uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería en Tecnologías Industriales
2017-2018

Trabajo Fin de Grado

"Diagramas PM aplicados a Sistemas Discretos"

Marcelo Payán Gardelegui

Tutor

Luis Santiago Garrido Bullón

Leganés, 2018

RESUMEN

El objetivo del presente archivo es documentar el trabajo realizado en la adaptación de un generador de controladores PID para sistemas continuos, realizado en Matlab por el Departamento de Ingeniería de Sistemas y Automática de la Universidad Carlos III de Madrid, para la generación de controladores PID para sistemas discretos.

Dicho generador permite también el análisis del funcionamiento del controlador mediante diagramas PM y de Bode, además de la generación de controladores Lead y Lag Network para ofrecer un control robusto.

La herramienta adaptada para controladores discretos incluye además la posibilidad del cálculo de controladores Lead-Lag Network y de controladores PID realizados utilizando el algoritmo de optimización Differential Evolution (DE).

Palabras clave: Ingeniería de control, Lugar de las raíces, Diagrama de Bode, Control PID, Control robusto, Control óptimo, Margen de ganancia, Margen de fase.

ABSTRACT

The objective of this file is to document the work done in the adaptation of a PID controller's generator for continuous systems, made in Matlab by the Systems Engineering and Automation Department of the Universidad Carlos III de Madrid for the generation of PID controllers for discrete systems.

This generator also allows the analysis of the controller operation by means of PM and Bode diagrams, as well as the generation of Lead and Lag Network controllers in order to offer a robust control.

The adapted tool for discrete controllers includes also the possibility of calculating Lead-Lag Network controllers and PID controllers using the Differential Evolution (DE) optimization algorithm.

Keywords: Control engineering, Root locus, Bode diagram, PID control, Robust control, Optimal control, Gain margin, Phase margin.

AGRADECIMIENTOS

A mi tutor Santiago Garrido por la idea de la que germinó este proyecto y por los consejos y orientación sin los cuales habría sido imposible llevarlo a buen término.

A la Universidad Carlos III de Madrid por permitirme convertirme en el profesional que siempre quise ser, por ofrecer todo lo que está a su disposición para facilitarme recorrer este camino.

A mis padres Inma y Juan por su apoyo incondicional durante toda mi recorrido académico y personal, por ser maestros en todas las facetas de la vida y por darme el regalo de sus valores.

A mi hermana Ángela por saber aguantarme en los momentos más difíciles de esta carrera y por ser la única capaz de hacerme ver mis errores, tú eres el orgullo de nuestra familia y estoy seguro de que tu vida será la que tú quieras que sea.

Por último, a mis amigos, a los que considero también mi familia, y a todos los que han compartido conmigo estos años tanto dentro como fuera de la universidad, un pedazo de este trabajo es también vuestro.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE FRAGMENTOS DE CÓDIGO

1. INTRODUCCIÓN

1.1. Motivación del trabajo

La concepción del trabajo nace de la necesidad del departamento de Ingeniería de Sistemas y Automática de la Universidad Carlos III de Madrid de desarrollar una herramienta para el cálculo y análisis de controladores PID en sistemas discretos que incorpore el análisis de sistemas utilizando los diagramas PM de fase y magnitud desarrollados por el propio departamento.

Puesto que previamente había sido desarrollada por dicho departamento una herramienta Matlab con la finalidad mencionada, aplicable en sistemas continuos, era posible adaptarla para su aplicación en sistemas discretos utilizando la teoría de la ingeniería de control de sistemas, adaptando también los diagramas PM.

Además de la adaptación de la herramienta, desde el inicio del planteamiento del trabajo se pretendió incluir también la posibilidad de generar controladores optimizados utilizando el algoritmo Differential Evolution (DE) que otorgasen mejores resultados que aquellos resultantes de aplicar métodos clásicos de la ingeniería de control como los criterios del módulo y el argumento.

1.2. Objetivos

El objetivo principal del trabajo es el desarrollo de la herramienta previamente mencionada, para lo cual se dividió el trabajo en 5 objetivos secundarios desarrollados posteriormente en cada uno de los epígrafes del capítulo 3 de este trabajo. Dichos objetivos secundarios son:

- Análisis de la herramienta del departamento para la generación de controladores continuos, con el fin de realizar una con un funcionamiento similar para permitir una adaptación sencilla de los métodos de trabajo aplicados en la primera.
- Diseño gráfico de la herramienta, utilizando para ello la utilidad GUIDE de Matlab.
- Desarrollo del código básico de la herramienta para la generación de controladores P, PI, PD y PID a partir de la función de transferencia del sistema a controlar y la selección de los polos que se desea que sean característicos del sistema controlado.
- Desarrollo del código para las opciones de generación de controladores Lead, Lag y Lead-Lag Network, o redes de adelanto, atraso y adelanto atraso, para un control robusto, a partir de la función de transferencia del sistema a controlar.

- Desarrollo del código para la opción de generación de controladores PID optimizados con el algoritmo Differential Evolution a partir de la función de transferencia del sistema a controlar.

1.3. Organización de la memoria

La memoria consta de dos partes principales, diferenciadas por la numeración de las páginas.

La parte previa al cuerpo del trabajo, numerada con números romanos, incluye la portada, el resumen o abstract del proyecto e índices de contenidos, figuras, tablas y fragmentos de códigos incluidos en la memoria, además de un apartado de agradecimientos personales.

La segunda parte, numerada con números arábigos, conforma el cuerpo del trabajo y está dividida en capítulos. El primero de ellos es una introducción en la que se explica la motivación (1.1) y objetivos del proyecto (1.2), además de la descripción del marco legal o regulador (1.4) que afecta al mismo.

El segundo de ellos está dedicado al estado de la cuestión y cuenta con un apartado dedicado a la historia de la ingeniería de control (2.1), describiendo los avances históricos que han conformado la base teórica de la misma, un segundo apartado en el que se analiza el estado del arte del control PID en concreto (2.2), y un último apartado en el que se expone la teoría referente a los diagramas PM utilizados en el proyecto (2.3).

El grueso del trabajo se incluye en el tercer capítulo, en el que se describen las soluciones adoptadas para cumplir los objetivos expuestos en el punto 1.2., seguido de un capítulo de conclusiones en el que se analizan el cumplimiento de estos objetivos (4.1), el impacto socio-económico del proyecto (4.2) y las nuevas líneas de trabajo que abre este proyecto (4.3) en los campos teórico y práctico.

Finalmente se incluyen un capítulo bibliográfico con las referencias utilizadas a lo largo de la memoria, referenciadas siguiendo el estilo IEEE, y un anexo en el que se incluyen los diagramas de flujo del cálculo de los controladores básicos, las redes de adelanto y atraso, y el controlador optimizado con DE.

1.4. Marco regulador

Las regulaciones que pueden afectar al problema solo son aplicables en el caso de que la aplicación Matlab fuese comercializada, no siendo este el objetivo del proyecto ya que el fin del mismo es académico y de investigación, como se explica en el punto 4.2 de este trabajo relativo al estudio económico del proyecto.

En caso de que la herramienta fuese a ser comercializada la primera normativa a la que estaría sujeta es a la impuesta por las licencias de Matlab, ya que la licencia académica con la que se realiza este proyecto no permite legalmente la comercialización del código desarrollado con la misma.

Puesto que la aplicación no recoge datos personales no está sujeta a las recientes normativas europeas sobre protección de datos, aunque sí a las referentes a propiedad intelectual y condiciones de contratación, pudiendo aplicar la siguiente normativa: [23]

- Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el Texto Refundido de la Ley de Propiedad Intelectual (en concreto los artículos 95 a 104, relativos a los programas de ordenador) sobre la protección jurídica de los programas de ordenador.
- Ley 7/1998, de 13 de abril, sobre Condiciones Generales de Contratación.
- Real Decreto Legislativo 1/2007, de 16 de noviembre, por el que se aprueba el texto refundido de la Ley General para la Defensa de los Consumidores y Usuarios y otras leyes complementarias.
- Ley 34/2002, de 11 de julio, de Servicios para lo Sociedad de la Información y comercio electrónico (en particular, los artículos 23 a 29 sobre contratación electrónica).
- El Art. 96 de la RDLeg. 1/1996 de 12 de Abr (TR. de la Ley de Propiedad Intelectual), respecto a los programas de ordenador

2. ESTADO DE LA CUESTIÓN

2.1. Historia de la ingeniería de control

Los primeras referencias a sistemas de control presentes en la historia se encuentran en la Antigua Grecia, entre las que destacan un reloj de agua o clepsidra diseñado por Ctesibio en el siglo III a.C. y utilizado posteriormente por Platón en la Academia de Atenas a modo de despertador para sus alumnos, un sistema de regulación del nivel de aceite de una lámpara diseñado por Filón de Bizancio también en el siglo III a.C., y diversos mecanismos descritos por Herón de Alejandría en sus obras *"Pneumática"* y "*Autómata"* del siglo I d.C, entre los que se incluyen diversos dispensadores de vino utilizando el principio de los vasos comunicantes y un odómetro para medir la distancia recorrida por un vehículo basándose en las revoluciones experimentadas por sus ruedas. [1]

Ya en la Edad Media, se comienzan a introducir sistemas de control en los molinos de viento encargados de moler grano, pudiéndose destacar cerca del año 1200 el diseño de H.U. Lansperg que controlaba la cantidad de grano suministrada para conseguir un funcionamiento óptimo, y posteriormente los diseños patentados en 1745 por E. Lee [2] y en 1787 por Thomas Mead [3]. El primero de ellos, considerado el primer servomecanismo de posición, controlaba la orientación e inclinación de las aspas del molino con el fin de aprovechar eficientemente la dirección del viento, el segundo controlaba tanto la presión ejercida por las piedras del molino como la variación del ángulo de las aspas del mismo, y sería el precursor del regulador centrífugo utilizado en las máquinas de vapor de Boulton y Watt, de gran importancia durante la Revolución Industrial a finales del siglo XVIII.

Basándose en dicho regulador, Meikle diseña un sistema propio para el control de los molinos de piedra, que sería utilizado a partir de 1788 por Watt para controlar la velocidad de su máquina de vapor, regulando la cantidad de vapor suministrado de la caldera a la turbina, constituyendo el primer sistema de la historia en incluir en el mismo sistema el sensor y el actuador sin un amplificador de potencia entre ellos. Los reguladores de Watt proporcionaban una acción proporcional sobre el sistema, permitiendo un control exacto solo con determinada carga mecánica y reduciendo su trabajo a únicamente un rango de velocidades. Posteriormente aparecerían reguladores con acción integral, como en los casos de los reguladores centrífugos para molinos diseñados por Willians Siemens en 1846 y 1853, en los que se sustituía la acción proporcional por acción integral, reduciendo así el offset del diseño de Watt. [1]

Durante el siglo XIX comienza el desarrollo de la Teoría de Control, enfocada al estudio de la estabilidad del sistema, que tiene sus antecedentes en la teoría de la variable compleja de Cauchy que completó la base matemática necesaria para la teoría y los trabajos de G.B. Airy en la regulación de telescopios [4], que permitió entender la influencia del amortiguamiento del sistema en su estabilidad. En 1868 Maxwell publica *"On Governors"* [5], que puede considerarse el origen de la Teoría de Control, ya que en él el autor expone que el comportamiento de un sistema de control cerca del equilibrio puede aproximarse usando una ecuación diferencial lineal, lo que permitía por lo tanto estudiar la estabilidad de un sistema, y por lo tanto la eficacia de su control, localizando las raíces de su ecuación característica. Ya en 1877, Routh publica su trabajo *"A treatise on the stability of a given state of motion"* [6] en el que presenta su criterio de estabilidad, usado hasta la actualidad para determinar la estabilidad de un sistema dinámico junto con el criterio equivalente de Hurwitz de 1885 [7].

En la línea del estudio de la estabilidad también destacan los trabajos de Heaviside en 1899 [8] sobre el análisis impusional en el estudio de sistemas dinámicos.
Estos dos criterios fueron usados para el diseño de controladores, junto con las transformaciones de Laplace y Fourier para determinar el modelo matemático del sistema, aproximadamente hasta 1932. A partir de este año aparecen tres importantes trabajos que conformarían la Teoría Clásica de Control, centrada en el estudio del sistema en el dominio de la frecuencia. Dichos trabajos son: *"Regeneration Theory"* de Nyquist en 1932 [9], en el que propone su criterio de estabilidad basado en la ganancia en lazo abierto del sistema, de gran utilidad práctica al ser esta directamente medible, *"Stabilized Feedback Amplifiers"* de Black en 1934 [10], que utiliza el criterio de Nyquist para el desarrollo de amplificadores realimentados, y *"Relations Between Attenuation and phase in Feedback Amplifier design"* de Bode en 1940 [11], en el que relaciona la amplitud y la fase del sistema en función de la frecuencia de la función de transferencia de la ganancia en lazo abierto, completando así el trabajo de Nyquist, introduciendo el concepto del margen de fase y ganancia del sistema y los diagramas logarítmicos de Bode.
Durante el siglo XX se produjeron diversos aportes que ayudaron a completar la teoría, como son los aportes de Evans en 1948 con *"Graphical Analysis of Control Systems"* [12] y en 1950 con *"Control System Synthesis by Root Locus Method"* [13], en los que presenta el método del lugar de las raíces para el estudio de un sistema, y de Truxal en 1954 con *"Feedback theory and control system synthesis"* [14], que incluye el Método del modelo, que permite determinar la función de transferencia que debe seguir el sistema de control a partir de las especificaciones deseadas. También se producen avances en lo referente al control discreto debido a la creciente importancia de los computadores, como lo son el cálculo de la transformada Z introducido por Salzer en 1954, y el criterio de Jury, incluido en su trabajo de 1958 *"Sampled data control Systems"* [15], y que permite determinar la estabilidad de un sistema discreto. En lo referente a los controladores PID, se debe destacar que a finales de los años 30 aparecen los primeros controladores neumáticos que utilizan las tres acciones proporcional, integral y derivativa, y la publicación en 1942 de *"Optimum Settings for Automatic Controllers"* [16] de Ziegler y Nichols, que ofrece un método práctico para el ajuste óptimo de un PID que sigue siendo profusamente utilizado en la actualidad.
A partir de 1955 se inicia el desarrollo de la Teoría Moderna de Control, caracterizada por la representación del sistema en variables de estado, con un trabajo casi exclusivo en el dominio del tiempo. Surgen los métodos de control óptimo, completados con el trabajo de Kalman en 1960 *"Contributions to the theory of optimal control"* [17], en el que se formulan los criterios de controlabilidad y observabilidad. Entre los años 60 y 70 se adopta un enfoque geométrico del problema de control, resolviéndolo utilizando métodos del álgebra lineal, se comienza a introducir el control adaptativo, con adaptación en términos de estructuras para la decisión, y se extiende el uso de computadores en procesos industriales, inicialmente para la supervisión de plantas de producción, y posteriormente para el control digital directo de procesos, gracias al impulso en eficiencia y la reducción en inversión que ofrecen a partir de los años 70 los microprocesadores. [1]

2.2. Situación actual del control PID

La tecnología utilizada en los controladores PID se ha mantenido en su fondo constante desde el comienzo de la utilización de los mismos en el ámbito del control industrial a finales de los años 30, poseyendo actualmente total vigencia como controladores robustos capaces de

generar lazos fiables de temperatura, presión, velocidad, posición, caudal y prácticamente cualquier variable del ámbito industrial a controlar, y constituyendo la primera opción de control para procesos desconocidos sobre los que no se puede obtener la información necesaria para generar un control óptimo, hasta el punto de que un 95 % de los lazos de control dentro de la industria son PID. [18]

El control PID se realiza por realimentación, calculando el error entre un valor medido por el sensor y el valor deseado, para luego enviar órdenes al actuador en función de tres parámetros: P (Proporcional), I (Integral) y D (Derivativo), que dan nombre al controlador. Dichos parámetros suelen ser explicados atendiendo al tiempo del controlador, que es analizado para calcular la acción de cada uno de ellos: el parámetro P está relacionado con el error actual presente en el sistema, el parámetro I valora el tiempo durante el que ha persistido dicho error, analizando de esta manera el pasado del controlador, y el parámetro D tiene en cuenta la tasa de variación del error, analizando de esta manera el futuro del controlador.

Utilizando ecuaciones, se pueden describir las acciones ejercidas por cada parámetro del controlador en función del error presente en ese momento en el sistema, que es la diferencia entre el valor medido por el sensor y el valor deseado por el usuario. De esta forma, al ser el control P únicamente proporcional al error medido, la acción realizada por el mismo es la expresada en la ecuación 2.1, siendo K_p la ganancia proporcional del controlador y e(t) el error medido en función del tiempo. Puesto que como se ha mencionado anteriormente el control I tiene en cuenta el tiempo que persiste el error, la acción que realiza es la expresada en la ecuación 2.2, siendo K_i la ganancia integral del controlador, e(τ) el error medido en cada instante de tiempo y τ la variable de integración que toma valores entre 0 y el presente t. Por último, en la ecuación 2.3 se expresa la acción realizada por el control D, que como se ha mencionado antes es calculada en función de la tasa de variación del error en función del tiempo, siendo K_d la ganancia derivativa del controlador y e(t) el error medido en función del tiempo. La acción total aplicada por el controlador PID será la suma de estas tres acciones. [19]

$$P = K_p e(t) \tag{2.1}$$

$$I = K_i \int_0^t e(\tau) d\tau \tag{2.2}$$

$$D = K_d \frac{de(t)}{dt} \tag{2.3}$$

Al ser un control regido únicamente por tres variables, K_p, K_i, y K_d, la labor del operario será únicamente la de seleccionar los valores asignados a cada una de ellas. Existen múltiples formas de realizar esto: si no se conoce el modelo matemático que rige el sistema a controlar es posible realizar un ajuste manual variando cada ganancia hasta obtener una salida del sistema apropiada o aplicar el método Ziegler-Nichols [16], basado en la ganancia proporcional a partir de la cual el sistema oscila, o incluso métodos más precisos como la optimización con software; por otro lado, modelando correctamente el sistema es posible obtener los valores necesarios para cumplir unos determinados requerimientos utilizando el Método del modelo de Truxal [14].

Los principales problemas que presenta el control PID son, además de que no garantiza un control óptimo ni la estabilidad del sistema, las limitaciones que presenta a la hora de controlar un sistema no lineal, debido a la linealidad del controlador, y la amplificación de ruido del sistema que se produce al incluir el parámetro derivativo, generando cambios bruscos en la salida. Dichos problemas han sido solucionados con variaciones en el algoritmo, además de introducir diversas mejoras en el control PID tradicional.

El problema de la linealidad puede resolverse con un control Feed-forward, que consiste en que el cálculo de la salida del controlador es realizado en lazo abierto utilizando el control PID en lazo cerrado para controlar el error del sistema, aumentando la precisión del control sin afectar la estabilidad del sistema ya que la salida del mismo no se ve afectada por el control en lazo cerrado. La solución del problema del ruido en controladores con parámetro derivativo generalmente consiste en eliminar dicho parámetro, resultando en un controlador PI más lento que el original, pero más estable.

Existen también modificaciones del algoritmo para problemas menos generales, como la variación Deadband que introduce una zona muerta en la salida del controlador para reducir así la frecuencia de activación del actuador en sistemas con mucha corrosión o desgaste, o la introducción de nuevas variables en los parámetros P y D para conseguir una mejor respuesta ante cambios bruscos del valor deseado. [19]

2.3. Teoría sobre diagramas PM

La idea de utilizar diagramas PM para el análisis de sistemas de control nace de la dificultad que presenta la representación de la función de transferencia de un sistema, ya que al ser una función compleja necesita 4 ejes para ser representada, algo difícil de imaginar al estar nuestro cerebro acostumbrado a trabajar en 3 dimensiones. Por esta razón a la hora de analizar un sistema de control suelen ser utilizadas varias representaciones de la función de transferencia simultáneamente, como pueden ser el lugar de las raíces y los diagramas de Bode, Nyquist y Nichols.

El diagrama PM, o diagrama de fase y magnitud, ofrece una solución a este problema codificando los valores de la fase del sistema con colores y la magnitud, en escala logarítmica, en curvas espaciadas igualmente, lo que provoca que el corte del diagrama PM

con el eje imaginario represente el diagrama de Bode del sistema. Esto permite leer visualmente de manera directa los márgenes de fase y ganancia, lo que ayuda al usuario a detectar características que debe poseer el sistema de control únicamente mirando el diagrama. Además, puesto que el uso de colores representa la fase entre 0 y 180º, la línea cian, que representa esos 180º, es el lugar de las raíces del sistema, mientras que la línea roja, representando 0º, es el lugar de las raíces inverso del sistema. [20]

La forma de calcular dichos márgenes está representada en la figura 2.1, que representa el diagrama PM de la función de transferencia en la ecuación 2.4, cuyos polos en bucle cerrado aparecen señalizados con cuadrados rojos. Como se encuentra indicado en la figura, cada división entre las curvas de magnitud equivale a 2 dB de magnitud, y cada división entre las curvas de fase equivale a 20º de fase. Además, incluye los verdaderos valores de los márgenes de ganancia y fase, 9.57 dB el primero, y 96.7º el segundo.

$$G(s) = \frac{1}{(s+1)(s+2)(s+3)} \tag{2.4}$$

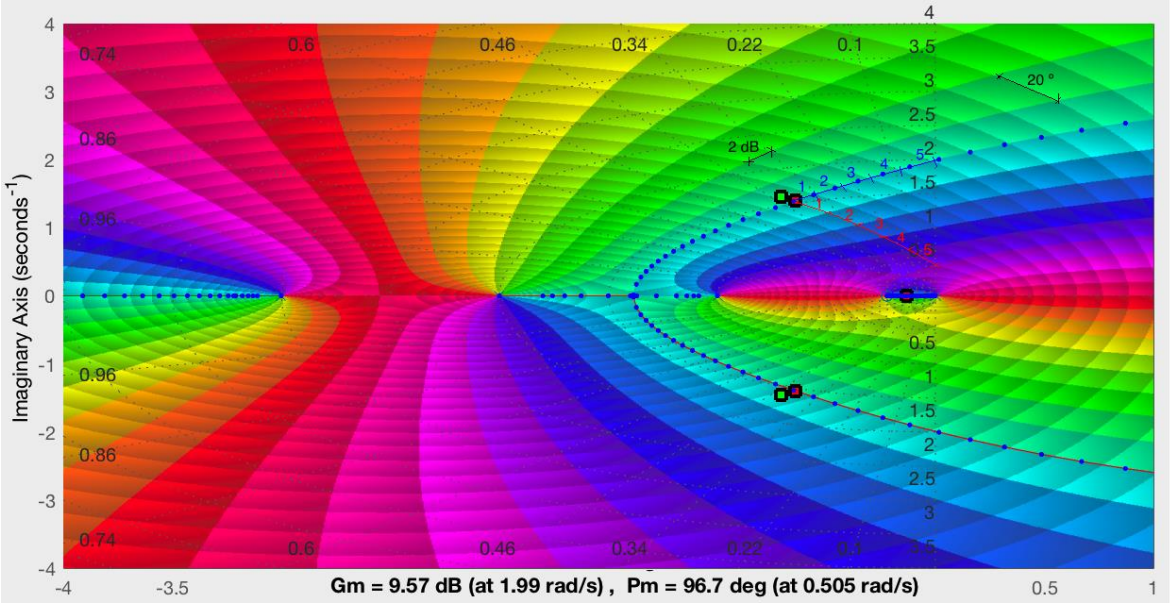

Figura 2.1. Diagrama PM de la función de transferencia en la ecuación 2.4

Para medir el margen de ganancia del sistema se debe seguir la línea cian, que representa el lugar de las raíces, desde los polos en bucle cerrado hasta el corte con el eje imaginario, midiendo después el número de divisiones entre curvas de magnitud atravesadas en ese segmento, como está representado en la línea de color azul oscuro. Se contabilizan 5 divisiones entre curvas de magnitud que, al equivaler cada una a 2 dB, suponen que el margen de ganancia del sistema es de aproximadamente 10 dB, muy cercano al valor real.

Para medir el margen de fase del sistema se debe seguir la curva de magnitud que atraviesa los polos en bucle cerrado desde estos al corte con el eje imaginario, midiendo después el número de divisiones entre curvas de fase atravesadas en ese segmento, como está representado en la línea de color rojo. Son contabilizadas 5 divisiones entre curvas de fase que, al equivaler cada una a 20º, suponen que el margen de fase del sistema es de aproximadamente 100º, muy cercano al valor real.

En el caso de representar la función de transferencia de sistemas discretos, se incluye una malla equivalente a la circunferencia unitaria, que permite calcular los márgenes de ganancia y fase del sistema utilizando las mismas pautas que en sistemas continuos, con la variación de que las mediciones se realizan hasta la circunferencia unitaria en lugar de hasta el eje imaginario. [20]

Esto está representado en la figura 2.2, que representa el diagrama PM de la función de transferencia en la ecuación 2.5, además de los valores reales de los márgenes de ganancia y fase del sistema, 6.02 dB y 46.6º respectivamente.

$$G(z) = \frac{1}{z(z-1)} \tag{2.5}$$

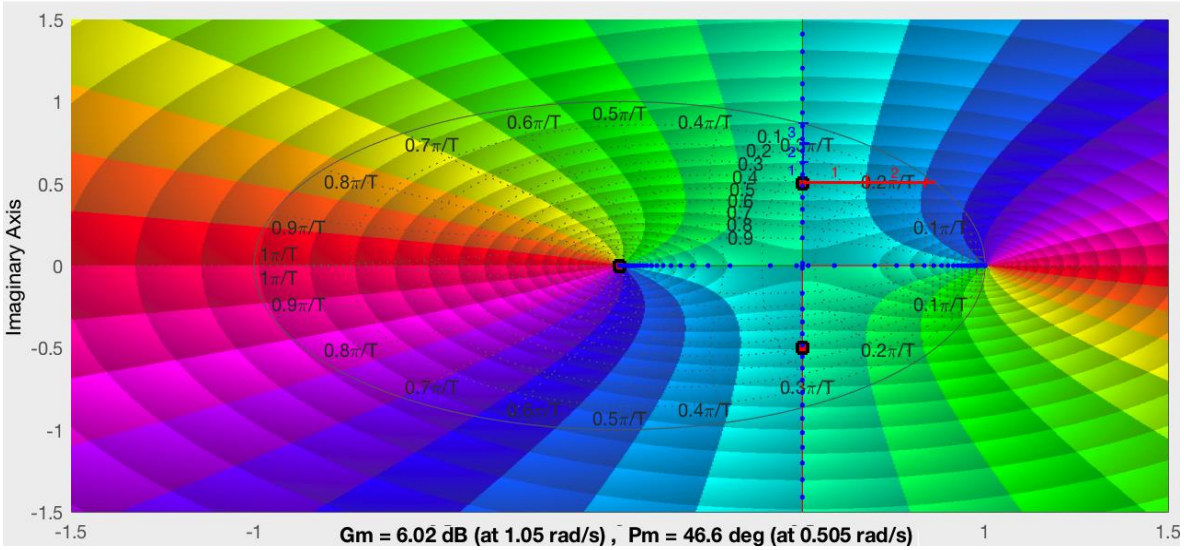

Figura 2.2. Diagrama PM de la función de transferencia en la ecuación 2.5

Observando la línea de color azul oscuro, se contabilizan 3 divisiones entre curvas de magnitud, lo que supone un margen de ganancia de aproximadamente 6 dB, muy cercano al valor real; mientras que observando la línea de color rojo, se contabilizan aproximadamente 2.3 divisiones entre curvas de fase, lo que supone un margen de fase de aproximadamente 46º, muy cercano al valor real.

3. DESARROLLO DE LA SOLUCIÓN

3.1. Análisis de la herramienta para sistemas continuos

La aplicación de generación de controladores para sistemas continuos es una interfaz gráfica de usuario (GUI) de Matlab, diseñada y desarrollada por el departamento de Ingeniería de Sistemas y Automática de la Universidad Carlos III de Madrid utilizando el entorno de desarrollo de GUI (GUIDE) que ofrece Matlab para el desarrollo de interfaces gráficas de manera interactiva. Al crear una aplicación de esta forma Matlab genera dos archivos automáticamente, un archivo de extensión .fig que contiene la información sobre el apartado gráfico de la misma, en este caso PMGUIbig.fig, y uno de extensión .m que contiene el código ejecutado por la herramienta, llamado en este caso PMGUIbig.m.

La interfaz gráfica en el momento de iniciarla puede verse en la figura 3.1, en la que aparecen identificados los diferentes componentes de la misma para facilitar su explicación. Los elementos text son o bien fragmentos de texto predefinidos como en los elementos text1 y text3, o cuadros de texto rellenados con el desarrollo del programa, como en el caso de los elementos text2, en el que se mostrará la función de transferencia del controlador calculado, text4 y text5, en los que se mostrarán los márgenes de fase y ganancia del sistema antes y después de la acción de control, respectivamente. Los elementos axes son ejes en los que se representarán tanto los diagramas PM, en axes1 y axes2, como los diagramas de Bode, en axes3 y axes4. Por último, los elementos edit son cuadros de texto editables por el usuario al ejecutar el programa, en este caso en edit1 se introduce la función de transferencia del sistema a controlar y en edit2 los límites de los ejes x e y de los diagramas PM en los que serán representados los sistemas. Además se indica donde se encuentran tanto los botones para la selección del tipo de controlador deseado como botones para seleccionar opciones extra de análisis.

La función de cada elemento en el uso de la aplicación se detalla a continuación en la descripción del código, que ha sido dividido en 5 partes: la inicialización de la aplicación, la representación del sistema a controlar, las funciones auxiliares utilizadas para el punto anterior, el cálculo de cada uno de los tipos de controladores y las opciones de análisis extra.

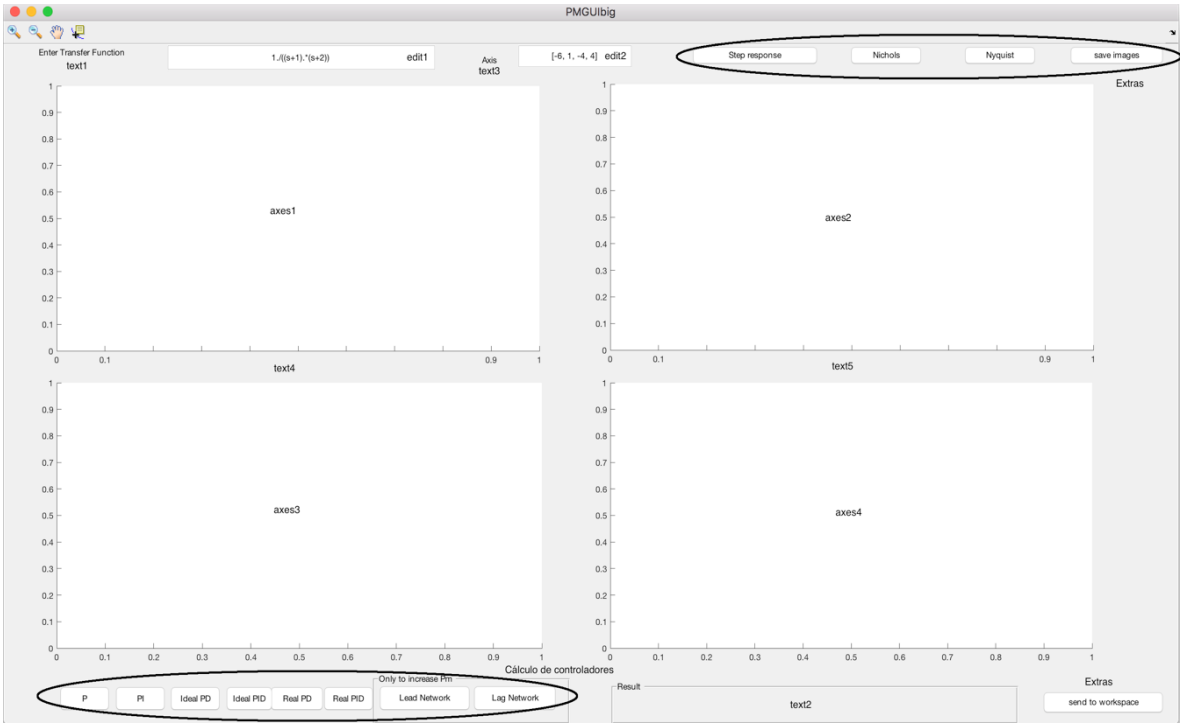

Figura 3.1. Interfaz gráfica de la herramienta para sistemas continuos al inicializarse

Inicialización

El código de inicialización se encuentra en la función PMGUIbig que da nombre a la aplicación, en el fragmento de código 3.1, y se genera automáticamente al crear la herramienta utilizando GUIDE; se explicita que no debe ser editado y con él se genera la estructura de la herramienta antes de hacerse visible al usuario.

Además en esta parte del código se incluyen las funciones PMGUIbig_OpeningFcn y PMGUIbig_OutputFcn que conectan la consola gráfica con el código Matlab para permitir interactuar con la misma por medio de entradas y salidas, también generadas automáticamente al crear la herramienta. Por otro lado se generan tres variables globales introducidas por el desarrollador de la herramienta: sys, sys2 y controller, cuya utilidad es explicada en los siguientes apartados. Estas dos funciones junto con las variables globales generadas se encuentran en el fragmento de código 3.2.

```
function varargout = PMGUIbig(varargin)
%PMGUIbig M-file for PMGUIbig.fig
%      PMGUIbig, by itself, creates a new PMGUIbig or raises the existing
%      singleton*.
%
%      H = PMGUIbig returns the handle to a new PMGUIbig or the handle to
%      the existing singleton*.
%
%      PMGUIbig('Property','Value',...) creates a new PMGUIbig using the
%      given property value pairs. Unrecognized properties are passed via
%      varargin to PMGUIbig_OpeningFcn.  This calling syntax produces a
%      warning when there is an existing singleton*.
%
%      PMGUIbig('CALLBACK') and PMGUIbig('CALLBACK',hObject,...) call the
%      local function named CALLBACK in PMGUIbig.M with the given input
%      arguments.
%
%       *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help PMGUIbig

% Last Modified by GUIDE v2.5 16-Mar-2014 08:54:03

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @PMGUIbig_OpeningFcn, ...
                   'gui_OutputFcn',  @PMGUIbig_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

Fragmento de código 3.1. Inicialización de PMGUIbig

```
% --- Executes just before PMGUIbig is made visible.
function PMGUIbig_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from the
%            command line (see VARARGIN)

% Choose default command line output for PMGUIbig
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
global sys sys2 controller

% UIWAIT makes PMGUIbig wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = PMGUIbig_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

Fragmento de código 3.2. Funciones de inicialización y variables globales de PMGUIbig

Representación del sistema a controlar

Para la representación de la función de transferencia del sistema a controlar es necesario tanto la propia función, introducida en el cuadro de texto edit1, como los límites de los ejes x e y en los que el usuario desea que sea representada, introducidos en el cuadro de texto edit2, por lo tanto el código para su representación se encuentra duplicado en las funciones que GUIDE genera para dichos cuadros de texto, y que se activan al pulsar ENTER en el interior de cualquiera de ambos cuadros. Dichas funciones se llaman edit1_Callback, para el cuadro de texto edit1, y edit2_Callback, para el cuadro de texto edit2, mostrándose aquí únicamente la primera de ellas, ya que su contenido es idéntico a excepción del nombre de la función, distribuida en los fragmentos de código 3.3, 3.4 y 3.5.

En el código de dicha función primero se almacenan los datos introducidos por el usuario, en la variable local ejes se almacenan los límites de los ejes y en sfun se almacena la función de transferencia del sistema, asociando posteriormente la variable local hax1 a los ejes axes1 de la consola gráfica e inicializándolos. Para representar el diagrama PM de la función de transferencia se utiliza la función auxiliar PMdiagrFunction que será explicada

posteriormente, introduciendo como entradas las variables antes mencionadas hax1, zfun (el mismo texto que sfun sustituyendo cada ‘s’ por ‘z’) y ejes.

```
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as
a double
global sys sys2 controller

StringAxis=get(handles.edit2,'String');
ejes=str2num(StringAxis);
sfun=get(handles.edit1,'String');
zfun = strrep(sfun, 's', 'z');
sfun = strrep(sfun, './', '/');
newsfun=strrep(sfun,'.*','*');
hax1=handles.axes1;
%set(hax1,'linewidth',1)
cla(hax1,'reset');
hold(hax1,'off');
axis(hax1,'off')
axis(hax1,'on')
PP=PMdiagrFunction(hax1,zfun,ejes);
```

Fragmento de código 3.3. Diagrama PM del sistema a controlar en PMGUIbig

A continuación se almacena en la variable global sys la función de transferencia del sistema y se representa sobre el diagrama PM el lugar de las raíces del sistema, además de los polos en bucle cerrado del mismo. También se establece la interacción con el diagrama por medio del cursor utilizando la función auxiliar NewCallback, que será explicada posteriormente.

Finalmente se asocia la variable local hax3 a los ejes axes3 de la consola gráfica y se representa en ellos el diagrama de Bode del sistema, escribiendo además en el cuadro de texto text4 los márgenes de fase y ganancia del mismo.

```
%showaxes('boxoff');
s=tf('s');
sys=eval(newsfun);
hold(hax1,'on');
%axis(hax1,'off')
h1=rlocusplot(hax1,sys);
p1=getoptions(h1);
p1.Title.String='';
p1.Xlabel.String='       ';
p1.XLabel.Color='w';
setoptions(h1,p1);
axis(ejes);
dcm_obj = datacursormode;
set(dcm_obj,'UpdateFcn',@NewCallback)

sysc=feedback(sys,1);
pc=pole(sysc);
hold(hax1, 'on');
for ii=1:length(pc)
    plot(hax1,real(pc(ii)),imag(pc(ii)),'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','m','MarkerSi
ze',7)
end
r=rlocus(sys);
plot(hax1,real(r),imag(r),'b.');
```

Fragmento de código 3.4. Lugar de las raíces del sistema a controlar e interacción con el diagrama en PMGUIbig

```
hax3=handles.axes3;
axis(hax3,'off')
axis(hax3,'on')
h3=bodeplot(hax3,sys);
[Gm,Pm]=margin(sys);
set(handles.text4,'String',strcat('Pm=',num2str(Pm),',
Gm=',num2str(Gm)));
hold on
margin(sys);
%title('');
xlabel('');
hold off
drawnow
```

Fragmento de código 3.5. Diagrama de Bode y márgenes de fase y ganancia del sistema a controlar en PMGUIbig

Funciones auxiliares

La función auxiliar PMdiagrFunction, en el fragmento de código 3.6, representa un diagrama PM utilizando tres entradas: hax, con la referencia al sistema de ejes en los que se representará el diagrama, fun, con la función de transferencia a representar, y ejes, con los límites de los ejes en los que será representado el diagrama.

```
function [PP]=PMdiagrFunction(hax,fun,ejes)

%ejes=([-6,1,-4,4]);
axis(hax,[ejes,0,100]);
xres = 800;
yres = 800;

z = zdomain(ejes(1)+ejes(3)*1i,ejes(2)+ejes(4)*1i,xres,yres);
%z = zdomain(-6-4*1i,1+4*1i,xres,yres);

f=eval(fun);
height=20*log10(abs(f))-100; hmin =-1000; hmax=1000;
pres=18;
t=exp(1i*linspace(-pi,pi,pres+1)); t=t(1:pres);
pres = length(t);
RGB = colscheme(f,t,pres);

PP = surf(hax,(real(z)),(imag(z)),height,RGB);
set(PP,'EdgeColor','none');
ax = axis;
axis(hax,[ax,hmin,hmax]);
view(hax,0,90)
axis(hax,'tight')
```

Fragmento de código 3.6. Función auxiliar PMdiagrFunction

Con su código se genera una superficie 3D representada en el sistema de ejes introducido en la variable hax, en la que los límites de los ejes x e y son los introducidos en la variable ejes y en la que la variable z es representada la función de transferencia del sistema introducida en la variable fun en escala logarítmica, representando la magnitud del sistema. Para la representación de la fase se utiliza un sistema de colores RGB generado con la función colscheme, en el fragmento de código 3.8, que codifica la fase del sistema en 18 niveles y la distribuye en franjas circularmente, coloreando los ceros del sistema en negro y los polos en blanco. Finalmente cambia la vista del diagrama a dos dimensiones resultando en el tipo de diagramas PM descritos en el punto 2.3 de este trabajo. La función zdomain no es comentada ya que su función es detectar errores del usuario al introducir datos.

La función auxiliar NewCallback, en el fragmento de código 3.7, genera un cuadro de texto con los valores de x e y en la posición que se ha pulsado con el cursor, además de los valores de magnitud y fase de la posición pulsada en caso de que exista también una coordenada z, es decir, que la posición seleccionada corresponda a una función de transferencia representada. Puesto que la función no recibe la función de transferencia del sistema a

controlar, el valor de la fase mostrado solo será válido para la función de transferencia por defecto en edit1.

```
function output_txt = myfunction(obj,event_obj)
% Display the position of the data cursor
% obj          Currently not used (empty)
% event_obj    Handle to event object
% output_txt   Data cursor text string (string or cell array of strings).

pos = get(event_obj,'Position');
x=pos(1);
y=pos(2);
s=x+y*1j;


f=1./(s+1)./(s+2);


ang=360-((angle(-f)+pi)*180/pi);

output_txt = {['X: ',num2str(pos(1),4)],...
    ['Y: ',num2str(pos(2),4)]};

% If there is a Z-coordinate in the position, display it as well
if length(pos) > 2
    output_txt{end+1} = ['Mag: ',num2str(pos(3)+100,4)];
    output_txt{end+1} = ['Phase:',num2str(ang,4)];
end
```

Fragmento de código 3.7. Función auxiliar NewCallback

```
function rgb = colscheme(f,t,pres)


t=exp(1i*linspace(-pi,pi,21)); t=t(1:20);
phaseres = 18;
% number of enhanced isochromatic lines

[m,n]=size(f);
% encoding phase

colmap=hsv(pres);
palette = colormap(colmap);

% number of colors in palette encoding phase

p = size(palette);
p = p(1);

% encoding phase

myangle = (angle(-f)+pi)/(2*pi);
nphase  = stepfct(myangle,p);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

%% phase plot with conformal polar grid

   blackp = sawfct(myangle,1/phaseres,0.75,1);
   %blackm = sawfct(2*log(abs(f)),2*pi/phaseres,0.75,1);
   blackm = sawfct(2*log(abs(f)),2*0.225,0.75,1);

   black = blackp.*blackm;

   rgb(:,:,1) =  black.*reshape(palette(nphase,1),m,n);
   rgb(:,:,2) =  black.*reshape(palette(nphase,2),m,n);
   rgb(:,:,3) =  black.*reshape(palette(nphase,3),m,n);

   rgb = BrightenRGB(rgb,.2);


%%        coloring        zeros        black,        poles        white
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

rgb(:,:,1) =  rgb(:,:,1) .* (abs(f)>0) + (1-rgb(:,:,1)).* (f==Inf);
rgb(:,:,2) =  rgb(:,:,2) .* (abs(f)>0) + (1-rgb(:,:,2)).* (f==Inf);
rgb(:,:,3) =  rgb(:,:,3) .* (abs(f)>0) + (1-rgb(:,:,3)).* (f==Inf);
```

Fragmento de código 3.8. Función auxiliar colscheme.

Cálculo de controladores

El código que calcula cada uno de los diferentes controladores que ofrece la herramienta, que se ejecuta al pulsar en alguno de los botones de controladores, posee una parte común a todos ellos referente a la inicialización del cálculo y a la presentación de resultados, y una parte que difiere en función del tipo de controlador seleccionado.

La parte común del código se encuentra dividida en los fragmentos de código 3.9, 3.10 y 3.11. En el primero de ellos (3.9) se leen tanto la función de transferencia del sistema a controlar como los límites de los ejes en los que se desea representar el sistema, se almacena en la variable global sys dicha función de transferencia y se almacenan en la variable local pc los polos del sistema en bucle cerrado. Además se pide al usuario que seleccione con el cursor sobre el diagrama PM del sistema a controlar la localización de los polos que se desean sean característicos del sistema controlado, exceptuando el caso del controlador PI ya que comúnmente dicho tipo de controlador se utiliza para eliminar el error en régimen estacionario del sistema, careciendo generalmente de importancia la localización de los polos característicos del sistema controlado.

Entre los fragmentos 3.9 y 3.10 se calcula el controlador seleccionado, almacenando la función de transferencia del controlador en la variable global controller y la del sistema controlado en la variable global sys2.

En el fragmento 3.10 se representa en el sistema de ejes axes2, identificado con la variable local hax2, el diagrama PM del sistema controlado, representando además sobre él el lugar de las raíces del sistema controlado, junto con los polos que se seleccionaron como deseados, en color verde, y los polos característicos del nuevo sistema, en color rojo.

Finalmente, en el fragmento 3.11 se representa en el sistema de ejes axes4, utilizando la variable local hax4, el diagrama de Bode del sistema controlado, y se escriben tanto los márgenes de fase y ganancia del sistema controlado como la función de transferencia del controlador en los cuadros de texto text5 y text2 respectivamente.

```
global sys sys2 controller

StringAxis=get(handles.edit2,'String');
ejes=str2num(StringAxis);
hax1=handles.axes1;

sfun=get(handles.edit1,'String');
zfun = strrep(sfun, 's', 'z');

sfun = strrep(sfun, './', '/');
newsfun=strrep(sfun,'.*','*');
s=tf('s');
sys=eval(newsfun);
sysc=feedback(sys,1);
pc=pole(sysc);
for ii=1:length(pc)
    plot(hax1,real(pc(ii)),imag(pc(ii)),'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','m','MarkerSiz
e',7)
end

[x,y]=ginput(1);
plot(hax1,x,y,'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerSiz
e',7)
plot(hax1,x,-y,'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerSiz
e',7)
drawnow
oldpos=pc(1);
z=oldpos;
foldpos=eval(zfun);
newpos=x+y*1j;
z=newpos;
fnewpos=eval(zfun);
```

Fragmento de código 3.9. Inicialización del cálculo de controladores en PMGUIbig

```
hax2=handles.axes2;
cla(hax2,'reset');
hold(hax2,'off');
axis(hax2,'off')
axis(hax2,'on')
PP=PMdiagrFunction(hax2,newzfun,ejes);%with controller1
newsfun2 = strrep(newsfun2, './', '/');newsfun2 = strrep(newsfun2, '.*',
'*');
sys2=eval(newsfun2);%with controller1
hold(hax2,'on');
h2=rlocusplot(hax2,sys2);%with controller1
p2=getoptions(h2);
p2.Title.String='';
p2.Xlabel.String=strcat('                ');
p2.XLabel.Color='w';
setoptions(h2,p2);
axis(ejes);
sgrid;
drawnow
hold(hax2, 'on')
plot(hax2,x,y,'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerSiz
e',7)
plot(hax2,x,-y,'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerSiz
e',7)
grid on
grid off
sgrid
box(hax2,'on')
drawnow
r2=rlocus(sys2);%with controller1
ax = axis(hax2);
hold on
plot(hax2,real(r2),imag(r2),'b.')
axis(ax);
drawnow
```

Fragmento de código 3.10. Diagrama PM del sistema controlado en PMGUIbig

```
hax4=handles.axes4;
axis(hax4,'off')
axis(hax4,'on')
h4=bodeplot(hax4,Kc*sys2);
[Gm2,Pm2]=margin(Kc*sys2);
set(handles.text5,'String',strcat('Pm=',num2str(Pm2),',
Gm=',num2str(Gm2)));
hold on
margin(Kc*sys2);
%title('');
xlabel('');
hold off

Controller = strrep(Controller, './', '/');Controller = strrep(Controller,
'.*', '*');
set(handles.text2,'String',strcat('Controller=
',num2str(Kc),Controller));
controller=eval(Controller);
sys2=Kc*controller*sys;

pc2r=pole(feedback(sys2,1));
%hold(hax2, 'on')
for ii=1:length(pc2r)
    xx=real(pc2r(ii));yy=imag(pc2r(ii));
    if ax(1)<xx && xx<ax(2) && ax(3)<yy && yy<ax(4)
        plot(hax2,xx,yy,'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSiz
e',7)
    end
end
```

Fragmento de código 3.11. Controlador calculado, diagrama de Bode y márgenes de fase y ganancia del sistema controlado en PMGUIbig

Para el cálculo de cada uno de los controladores se calculan las posiciones de los polos y ceros en bucle cerrado de los mismos en función del tipo de controlador seleccionado, para después calcular la ganancia Kc con la ecuación 3.1, que evalúa el inverso de la función de transferencia realimentada del sistema controlado, representada como GH, en la posición de los polos deseados introducidos por el usuario, representados como s_{dp}. Este cálculo se encuentra en el fragmento de código 3.12.

$$Kc = \frac{1}{|GH|_{@s_{dp}}} \tag{3.1}$$

```
posgreen=newpos;
z=posgreen;%green
fposgreen=eval(newzfun);
sysc2=feedback(sys2,1);
pc2=pole(sysc2);
z=pc2(1);%blue
fposblue=eval(newzfun);
Kc=abs(fposblue)/abs(fposgreen);
```

Fragmento de código 3.12. Ganancia del sistema controlado en PMGUIbig

El cálculo de las posiciones de los polos y ceros del controlador varía en función del tipo de controlador seleccionado de la siguiente forma:

- Controlador P:

El controlador proporcional no introduce ningún polo o cero en el sistema, únicamente varía la ganancia Kc, por lo que para generarlo la herramienta calcula la ganancia Kc utilizando el código en el fragmento de código 3.12, siendo el controlador calculado igual a esta ganancia como se muestra en la ecuación 3.2.

$$controller = Kc \tag{3.2}$$

- Controlador PI:

El controlador integral suele ser utilizado para corregir el comportamiento del sistema en régimen estacionario eliminando el error, con independencia de la posición de los polos característicos del sistema controlado resultante. Por ello al seleccionar la opción de generar un controlador PI no se pide al usuario que introduzca la posición de los polos deseados, generando en cambio un controlador con ganancia unitaria y un polo en 0 como se muestra en la ecuación 3.3, donde ab es el opuesto de la posición de un cero del sistema controlado calculado utilizando la ecuación 3.4 en función de la posición de los polos en bucle cerrado del sistema antes de ser controlado, representados como pc, con una aproximación práctica a la hora de diseñar controladores PI que consiste en situar el cero a un sexto de la distancia del origen a la posición de los polos característicos del sistema, en el eje real. El cálculo de la posición de este cero se encuentra en el fragmento de código 3.13.

$$controller = \frac{s + ab}{s} \tag{3.3}$$

$$ab = \frac{|real(pc)|}{6} \tag{3.4}$$

```
ab=abs(real(pc(2)))/6;
```

Fragmento de código 3.13. Cero del controlador PI en PMGUIbig

- Controlador PD ideal:

El controlador derivativo teórico introduce un cero y divide la ganancia por la posición de dicho cero, tal como se muestra en la ecuación 3.5, siendo b esta posición, calculada utilizando el criterio del argumento con las expresiones 3.6 y 3.7. En la primera de ellas se calcula el ángulo que debe formar la nueva posición, en la ecuación DefAng, restando el ángulo de los polos característicos del sistema antes de ser controlado, en la ecuación pc, y el de los polos seleccionados como deseados por el usuario, en la ecuación pd, para luego en la segunda utilizando este ángulo calculado y la posición de los polos deseados obtener la posición b. Finalmente se calcula la magnitud de la ganancia Kc utilizando la ecuación 3.1 con la función de transferencia del sistema controlado. El cálculo de la posición b se encuentra en el fragmento de código 3.14, que se complementa con el fragmento de código 3.9.

$$controller = Kc\frac{(s-b)}{(-b)} \tag{3.5}$$

$$DefAng = angle(pc) - angle(pd) \tag{3.6}$$

$$b = real(pd) - \frac{imag(pd)}{tan(DefAng)} \tag{3.7}$$

```
beta=(angle(-foldpos)+pi)/(2*pi)*360;
alpha=(angle(-fnewpos)+pi)/(2*pi)*360;
DefAng=(beta-alpha);
b=x-y/tan(DefAng*pi/180);
```

Fragmento de código 3.14. Cero del controlador PD ideal en PMGUIbig

- Controlador PID ideal:

El control integral-derivativo teórico introduce en el sistema una combinación de los anteriores dos controladores, un polo en 0 junto con un cero a una distancia del origen de un sexto de la de los polos característicos del sistema sin controlar, calculado en la ecuación 3.4,

además de un cero en la misma posición b, calculada con el criterio del argumento en las ecuaciones 3.6 y 3.7, resultando finalmente el controlador de la ecuación 3.8 calculado en el fragmento de código 3.15, con la ganancia Kc dividida por el factor b, calculada utilizando la ecuación 3.1.

$$controller = Kc\frac{(s-b)}{(-b)}\frac{(s+ab)}{s} \tag{3.8}$$

```
beta=(angle(-foldpos)+pi)/(2*pi)*360;
alpha=(angle(-fnewpos)+pi)/(2*pi)*360;
DefAng=(beta-alpha);
b=x-y/tan(DefAng*pi/180);

ab=abs(real(pc(2)))/6;
```

Fragmento de código 3.15. Polo y ceros del controlador PID ideal en PMGUIbig

- Controlador PD real

El diseño del controlador derivativo en la práctica debe modificarse debido a la complejidad física de introducir un cero en el sistema sin compensarlo con un polo, como ocurre en el controlador PD ideal. Para su diseño se calculan dos factores, pole1 y zero1, que serán las posiciones del polo y el cero del controlador respectivamente, siendo además el cociente entre estos la ganancia del mismo, como se muestra en la ecuación 3.9. Para el cálculo de estos factores se requieren variables intermedias: a, en la ecuación 3.10, es el ángulo que forman los polos deseados seleccionados por el usuario, en la ecuación pd; DefAng, calculada utilizando la ecuación 3.6, y por último b y γ calculadas utilizando las ecuaciones 3.11 y 3.12 respectivamente. Finalmente, con las ecuaciones 3.13 y 3.14 se obtienen los factores pole1 y zero1 que permiten el diseño del controlador, en las que pd identifica a los polos deseados. El código que calcula estas variables y factores se encuentra en el fragmento de código 3.16.

$$controller = \frac{pole1}{zero1}\frac{(s-zero1)}{(s-pole1)} \tag{3.9}$$

$$a = angle(pd) \tag{3.10}$$

$$b = \frac{\frac{DefAng}{2}-a}{2} - \frac{DefAng}{2} \tag{3.11}$$

$$\gamma = DefAng + b \tag{3.12}$$

$$pole1 = real(pd) + imag(pd) \cdot tan(b) \tag{3.13}$$

$$zero1 = real(pd) + imag(pd) \cdot tan(\gamma) \tag{3.14}$$

```
beta=(angle(-foldpos)+pi)/(2*pi)*360;
alpha=(angle(-fnewpos)+pi)/(2*pi)*360;
DefAng=(beta-alpha);
phi=DefAng*pi/180;
a=atan2(y,x);
b=(phi/2-a)/2-phi/2;
pole1=x+y*tan(b);
gamma=phi+b;
zero1=x+y*tan(gamma);
```

Fragmento de código 3.16. Polo y cero del controlador PD real en PMGUIbig

- Controlador PID real:

El control integral-derivativo en la práctica presenta el mismo problema que el derivativo, por lo que se diseña un controlador como el de la ecuación 3.15, con los factores pole1 y zero1 calculados utilizando las ecuaciones 3.13 y 3.14 y la posición del cero ab utilizando la ecuación 3.4. El código con este cálculo se encuentra en el fragmento de código 3.17.

$$controller = \frac{pole1}{zero1}\frac{(s - zero1)}{(s - pole1)}\frac{(s + ab)}{s} \tag{3.15}$$

```
beta=(angle(-foldpos)+pi)/(2*pi)*360;
alpha=(angle(-fnewpos)+pi)/(2*pi)*360;
DefAng=(beta-alpha);
phi=DefAng*pi/180;
a=atan2(y,x);
b=(phi/2-a)/2-phi/2;
pole1=x+y*tan(b);
gamma=phi+b;
zero1=x+y*tan(gamma);

ab=abs(real(pc(2)))/6;
```

Fragmento de código 3.17. Polo y ceros del controlador PID real en PMGUIbig

El diseño y cálculo de las redes de adelanto y atraso (Lead Network y Lag Network) se realiza de forma muy similar en ambas herramientas, por lo que será explicado en el apartado 3.4 de

este trabajo cuando se trate su inclusión en el diseño de la herramienta para sistemas discretos.

Opciones de análisis extra

La herramienta ofrece cinco opciones extra, dos de ellas para almacenar resultados, y tres de ellas con opciones de análisis para complementar las que ofrecen los diagramas PM. Para el almacenamiento de resultados existen la opción send to workspace, cuyo código se incluye en el fragmento de código 3.18, que crea un archivo MATLAB de nombre data y escribe en él el contenido de las tres variables globales sys, con la función de transferencia del sistema a controlar, sys2, con la función de transferencia del sistema controlado, y controller, con la función de transferencia del controlador.; y la opción save images, cuyo código se incluye en el fragmento de código 3.19, que crea un archivo de imagen .bmp del conjunto de ejes axes2, es decir, una imagen del diagrama PM del sistema controlado.

```
%% Save Controller
% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global sys sys2 controller
%controller=eval(Controller);
assignin ('base','controller',controller);
assignin ('base','sys',sys);
assignin ('base','sys2',sys2);
save('data',sys,sys2,controller);
```

Fragmento de código 3.18. Opción send to workspace en PMGUIbig

Las otras tres opciones generan diferentes diagramas de análisis, y para ejemplificar su función se utilizará la función de transferencia de la ecuación 3.16 como sistema a controlar con un PID con polos deseados en aproximadamente 3 ± 2i, viéndose la consola gráfica de la herramienta como en la figura 3.2. Las opciones Step response, Nichols y Nyquist generan respectivamente la respuesta a entrada escalón, el diagrama de Nichols y el diagrama de Nyquist del sistema antes y después de ser controlado, lo que aparece ejemplificado con las figuras 3.3, 3.4 y 3.5 respectivamente. El código utilizado para cada una de estas opciones se encuentra en los fragmentos de códigos 3.20, 3.21 y 3.22, en el orden en el que han sido nombradas las opciones.

```
%% Save images
% --- Executes on button press in pushbutton14.
function pushbutton14_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
axes(handles.axes2);
[FileName, PathName] = uiputfile('*.bmp', 'Save As');
if PathName==0, return; end
Name = fullfile(PathName, FileName);
imwrite(handles.axes2, Name, 'bmp');
```

Fragmento de código 3.19. Opción save images en PMGUIbig

$$G(s) = \frac{1}{(s+1)(s+2)} \tag{3.16}$$

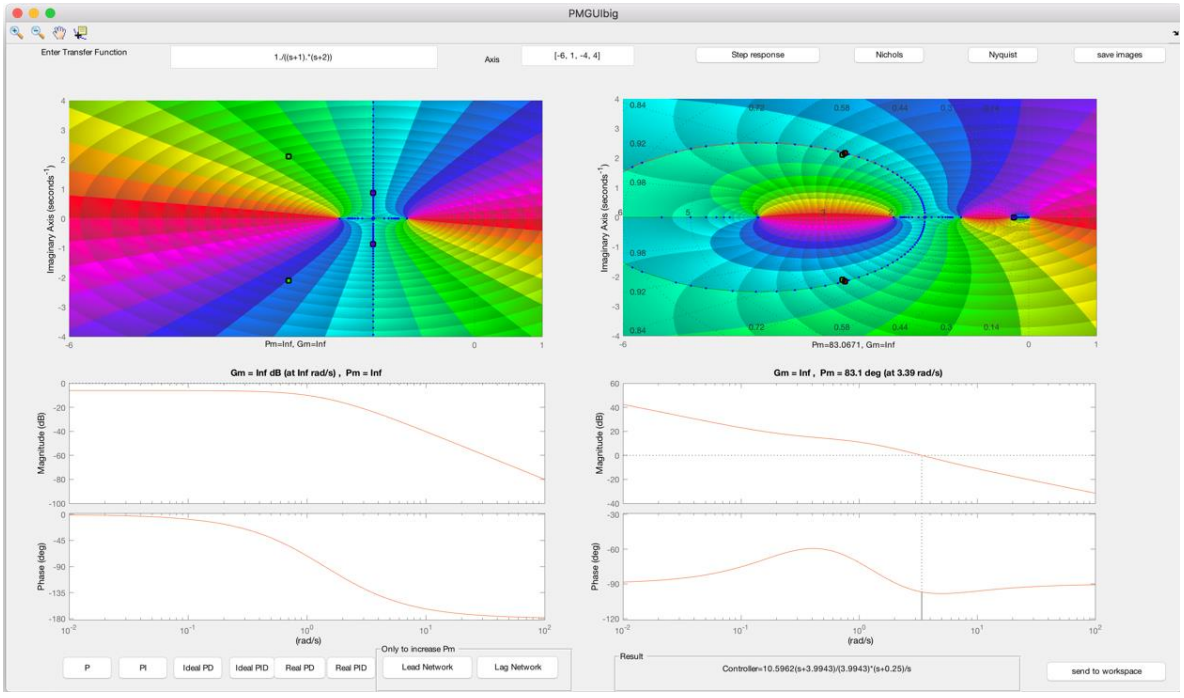

Figura 3.2. Interfaz gráfica de la herramienta para sistemas continuos con la función de transferencia en la ecuación 3.16 controlada

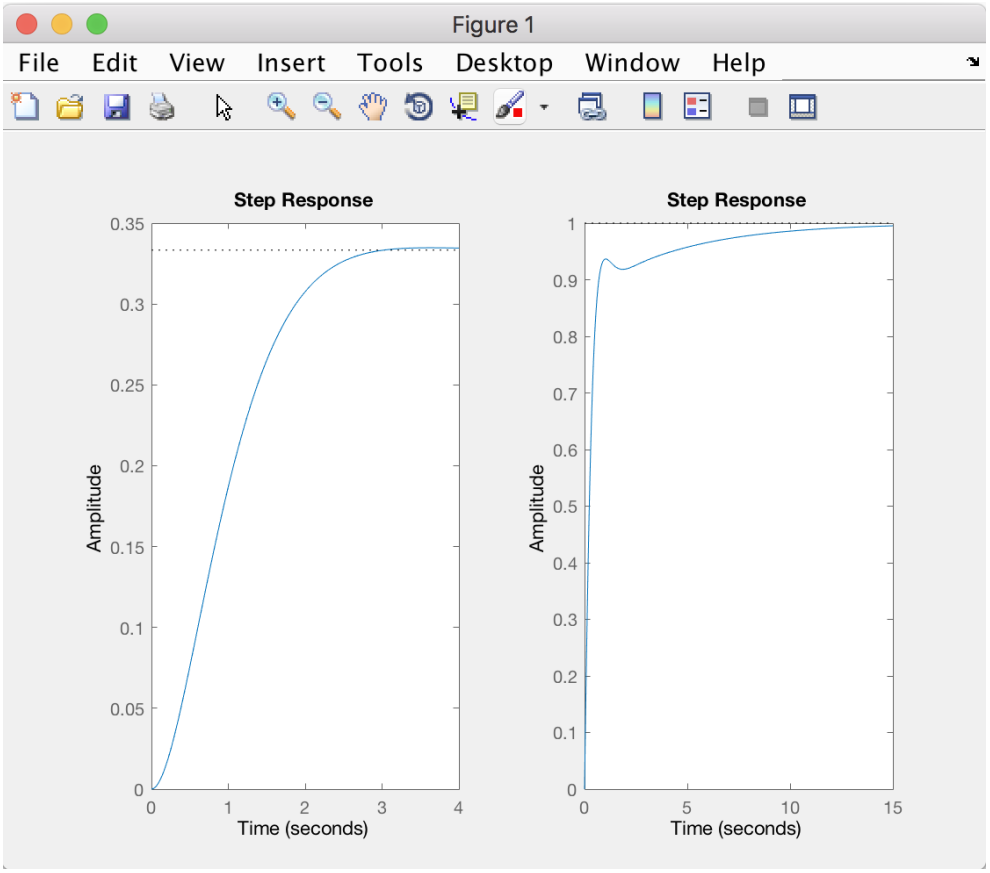

Figura 3.3. Ejemplo de la opción Step response

```
%% Step response
% --- Executes on button press in pushbutton11.
function pushbutton11_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global sys sys2 controller
figure
subplot(121)
step(feedback(sys,1))
subplot(1,2,2)
step(feedback(sys2,1))
```

Fragmento de código 3.20. Opción Step response en PMGUIbig

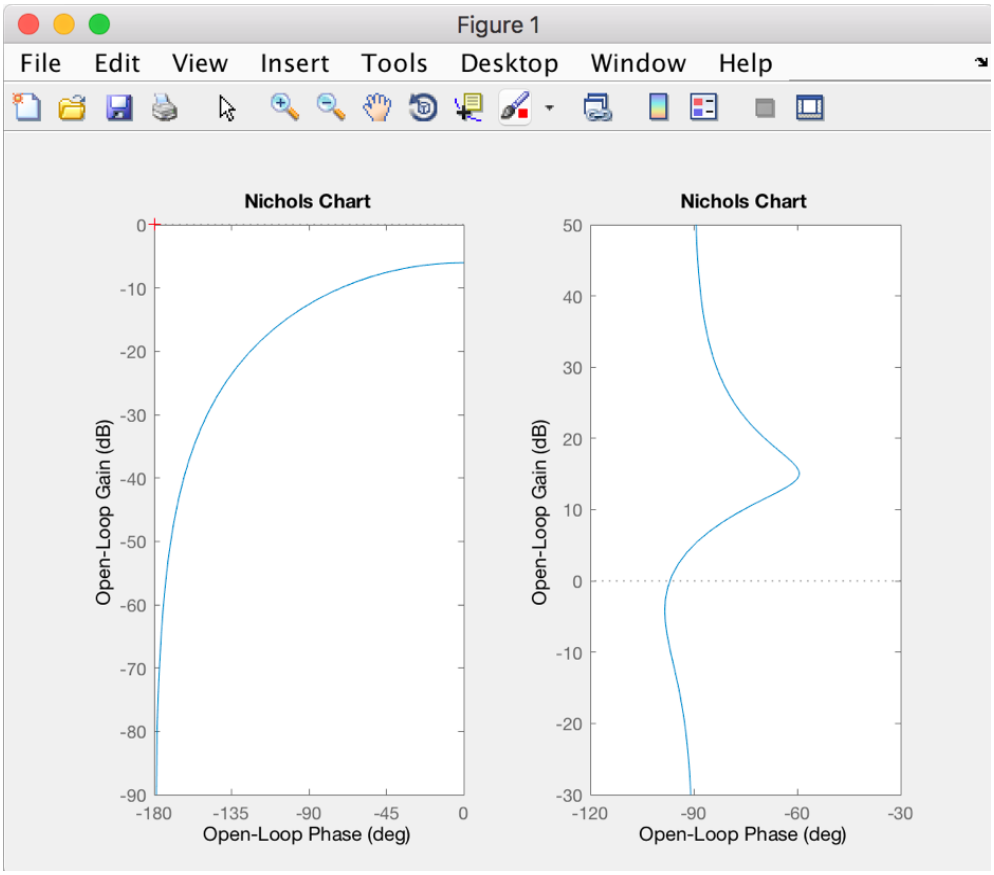

Figura 3.4. Ejemplo de la opción Nichols

```
%% Nichols
% --- Executes on button press in pushbutton12.
function pushbutton12_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global sys sys2 controller
figure
subplot(121)
nichols(sys)
subplot(1,2,2)
nichols(sys2)
```

Fragmento de código 3.21. Opción Nichols en PMGUIbig

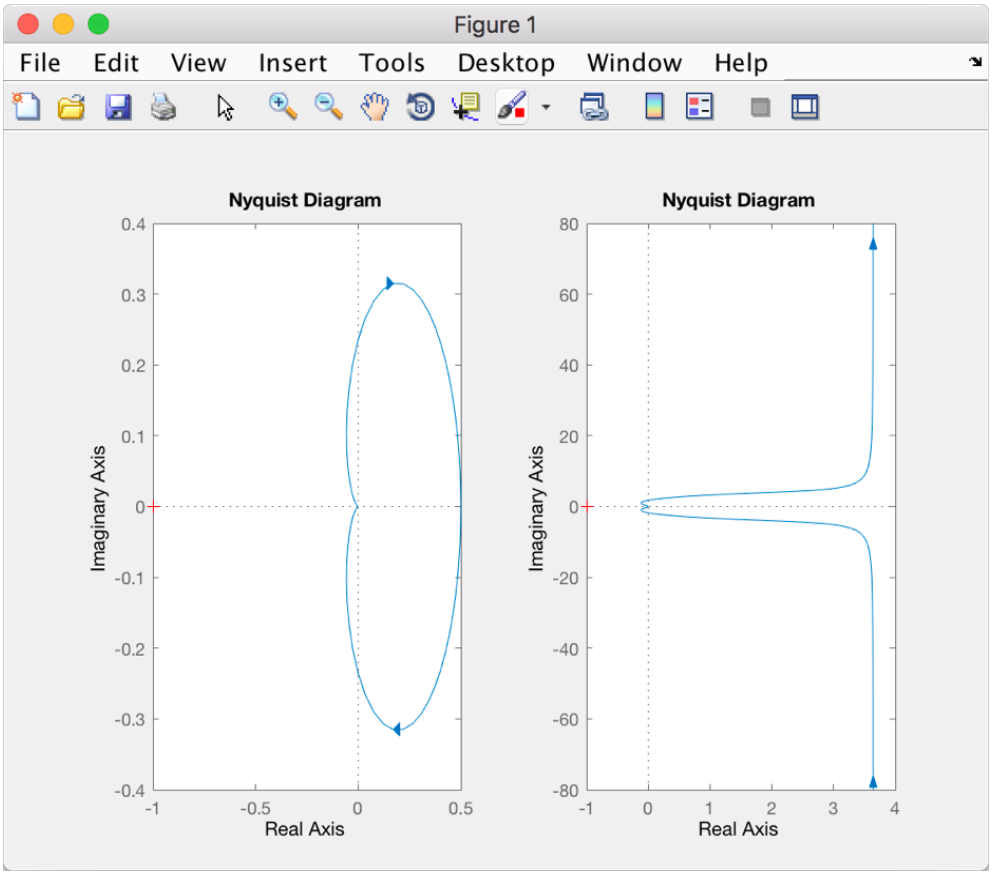

Figura 3.5. Ejemplo de la opción Nyquist

```
%% Nyquist
% --- Executes on button press in pushbutton13.
function pushbutton13_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global sys sys2 controller
figure
subplot(121)
nyquist(sys)
subplot(1,2,2)
nyquist(sys2)
```

Fragmento de código 3.22. Opción Nyquist en PMGUIbig

3.2. Diseño del apartado gráfico de la herramienta

La herramienta para generar controladores de sistemas discretos se diseña con el mismo método que se utilizó para diseñar la aplicable a sistemas continuos, utilizando el entorno de desarrollo GUIDE que ofrece Matlab para crear una interfaz gráfica de usuario. Al crear la interfaz con este método Matlab genera automáticamente los archivos PMGUIdiscbig.fig, que contiene la información gráfica de la interfaz y será editado usando el entorno GUIDE, y PMGUIdiscbig.m, que contiene el código de la aplicación que será explicado en los siguientes apartados de este trabajo.

La interfaz gráfica de la herramienta al iniciarla puede verse en la figura 3.6, en la que además se han identificado los componentes que la forman para facilitar su explicación. La mayoría de los componentes se encuentran también en la herramienta para sistemas continuos ya explicada en el apartado 3.1 de este trabajo, por lo que solo se explicarán los elementos cuyo nombre se ha modificado pero cumplen la misma función y aquellos que no aparecían en la herramienta anterior. De esta manera el fragmento de texto text3 pasa a llamarse text2 en esta herramienta, mientras que los cuadros de texto text2, text4 y text5 pasan a llamarse text6, text3 y text4 respectivamente, y los sistemas de ejes axes2 y axes3 pasan a llamarse axes3 y axes2 respectivamente.

Se incluye un selector de polos deseados con el fin de permitirle al usuario una mayor precisión en el momento de elegirlos en lugar de seleccionarlos directamente en el diagrama PM. Este selector está compuesto por el fragmento de texto predefinido text5 que indica el formato en el que se deben escribir los polos, y el cuadro de texto editable edit3 en el que se introducen los mencionados polos, permitiendo siempre seleccionarlos manualmente con el cursor sobre el diagrama PM en el caso de que el cuadro de texto edit3 no se edite.

Se eliminan las opciones de controladores PD y PID ideales al ser físicamente irrealizables en discreto, y se introducen las opciones de controlador PID optimizado utilizando el algoritmo Differential Evolution (DE) y de red Lead-Lag. Por último se incluyen los botones activables por el usuario pushbutton14, pushbutton15 y pushbutton 16 que permiten resetear a los valores por defecto los cuadros de texto edit1, edit2 y edit3 respectivamente.

En la representación de los diagramas PM la única diferencia con la anterior herramienta es la inclusión en los mismos de una malla representativa del circulo unitario, que permite estudiar la estabilidad del sistema de forma directa, lo cual se puede comprobar en la figura 3.7 que muestra el control PID, habiendo seleccionado como polos deseados 0.5 ± 0.5i, del sistema representado por la función de transferencia en la ecuación 3.17.

$$G(z) = \frac{0.1z}{(z-0.8)(z-0.9)} \tag{3.17}$$

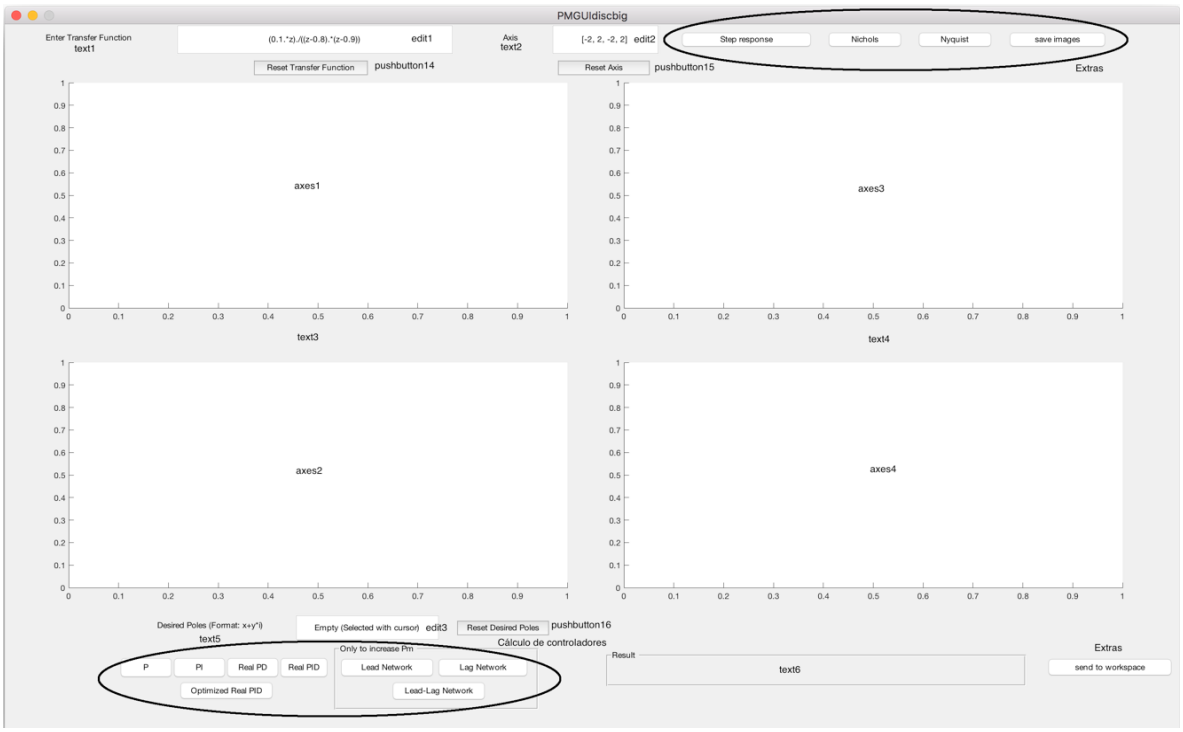

Figura 3.6. Interfaz gráfica de la herramienta para sistemas discretos al inicializarse

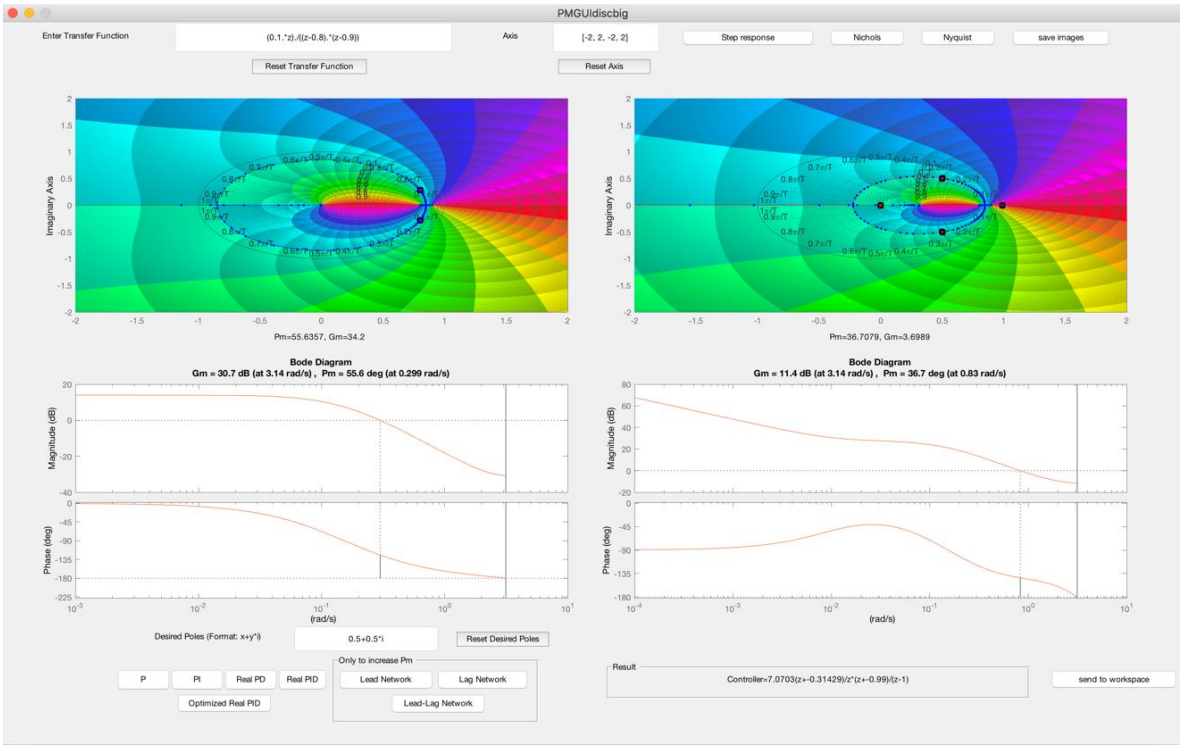

Figura 3.7. Interfaz gráfica de la herramienta para sistemas discretos con la función de transferencia en la ecuación 3.17 controlada

3.3. Desarrollo del código básico

Debido a que la herramienta para sistemas discretos está basada en el generador de controladores para sistemas continuos, su diseño gráfico y el desarrollo de su código en líneas generales es muy similar al de este, por lo que, exceptuando la parte del código referente al cálculo de controladores, gran parte del código ha sido reciclado de la anterior herramienta, realizando los cambios necesarios para emplear la herramienta para la generación de controladores discretos en lugar de continuos, trabajando por tanto en el plano discreto con la variable z en lugar de en el plano continuo con la variable s, además de los cambios relativos a los nombres de los elementos de la consola gráfica que cambian de una herramienta a otra.

Inicialización, representación del sistema a controlar y funciones auxiliares

La herramienta para sistemas discretos está desarrollada en el entorno GUIDE, al igual que la herramienta para sistemas continuos, lo que implica que las funciones de inicialización, en los fragmentos de código 3.1 y 3.2, se mantienen iguales en ambas herramientas incluida la inicialización de las tres variables globales, con la única diferencia de que en la herramienta discreta la función PMGUIbig se llama PMGUIdiscbig, PMGUIbig_OpeningFcn pasa a ser PMGUIdiscbig_OpeningFcn y PMGUIbig_OutputFcn pasa a llamarse PMGUIdiscbig_OutputFcn.

Al tener ambas herramientas una distribución de los elementos de la consola gráfica similar, la parte del código referente a la representación del sistema a controlar lo es también. De esta forma la representación del diagrama PM del sistema a controlar, para la herramienta de sistemas continuos en el fragmento de código 3.3, se mantiene igual en ambas herramientas, incluyendo un fragmento en el que se lee el valor del cuadro de texto edit3, se comprueba si se ha introducido un valor para los polos deseados o se desea seleccionar manualmente y en caso de que se haya introducido se almacena en la variable local DesiredPoles, las variables sfun y zfun intercambian sus nombres y la variable newsfun pasa a llamarse newzfun debido a que al trabajar con sistemas discretos que utilizan z como variable, la función de transferencia del sistema a controlar introducida en el cuadro de texto edit1 utilizará dicha variable. El código equivalente en la herramienta para sistemas discretos del fragmento de código 3.3 se encuentra en el fragmento de código 3.23.

La representación del lugar de las raíces del sistema se mantiene igual en ambas herramientas únicamente con los cambios referentes a la variable utilizada comentados en el párrafo anterior, sin embargo al establecer la interacción por cursor del usuario con el diagrama PM se utiliza la función NewCallback_discr, que será explicada posteriormente, en lugar de NewCallback y se añaden los comandos sgrid; y drawnow para la representación de la malla del círculo unitario comentada en el apartado 3.2. Esta parte del código, que para la herramienta para sistemas continuos se encuentra en el fragmento de código 3.4, se encuentra en el fragmento de código 3.24 para la herramienta para sistemas discretos.

```
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as
a double
global sys sys2 controller

StringAxis=get(handles.edit2,'String');
ejes=str2num(StringAxis);
StringPoles=get(handles.edit3,'String');
NoPoles='Empty (Selected with cursor)';
if strcmp(StringPoles,NoPoles)==1
    DesiredPoles=0;
else
    DesiredPoles=str2num(StringPoles);
end
zfun=get(handles.edit1,'String');
sfun = strrep(zfun, 'z', 's');
zfun = strrep(zfun, './', '/');
newzfun=strrep(zfun,'.*','*');
hax1=handles.axes1;
%set(hax1,'linewidth',1)
cla(hax1,'reset');
hold(hax1,'off');
axis(hax1,'off')
axis(hax1,'on')
PP=PMdiagrFunction(hax1,sfun,ejes);
```

Fragmento de código 3.23. Diagrama PM del sistema a controlar en PMGUIdiscbig

La representación del diagrama de Bode y el cálculo de los márgenes de fase y ganancia del sistema a controlar, en el fragmento de código 3.5 para la herramienta para sistemas continuos, sufre únicamente un cambio de variables debido a los cambios de los nombres de elementos producidos en el diseño de la consola gráfica. El conjunto de ejes axes3 en la herramienta discreta se llama axes2, por lo que ocurre lo mismo con las variables hax3 y h3 que pasan a llamarse hax2 y h2, y el cuadro de texto text4 pasa a llamarse text3 en la herramienta discreta. El código de la herramienta para sistemas discretos equivalente al fragmento de código 3.5 se encuentra en el fragmento de código 3.25. También se debe comentar que al haber un nuevo cuadro de texto en la consola gráfica, se genera también la función edit3_Callback, cuyo contenido será idéntico al explicado para edit1_Callback y edit2_Callback.

```
%showaxes('boxoff');
z=tf('z');
sys=eval(newzfun);
hold(hax1,'on');
%axis(hax1,'off')
h1=rlocusplot(hax1,sys);
p1=getoptions(h1);
p1.Title.String='';
p1.Xlabel.String='       ';
p1.XLabel.Color='w';
setoptions(h1,p1);
axis(ejes);
dcm_obj = datacursormode;
set(dcm_obj,'UpdateFcn', @NewCallback_discr)
sgrid;
drawnow

sysc=feedback(sys,1);
pc=pole(sysc);
hold(hax1, 'on');
for ii=1:length(pc)
    plot(hax1,real(pc(ii)),imag(pc(ii)),'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','m','MarkerSi
ze',7)
end
r=rlocus(sys);
plot(hax1,real(r),imag(r),'b.');
```

Fragmento de código 3.24. Lugar de las raíces del sistema a controlar e interacción con el diagrama en PMGUIdiscbig

```
hax2=handles.axes2;
axis(hax2,'off')
axis(hax2,'on')
h2=bodeplot(hax2,sys);
[Gm,Pm]=margin(sys);
set(handles.text3,'String',strcat('Pm=',num2str(Pm),',
Gm=',num2str(Gm)));
hold on
margin(sys);
%title('');
xlabel('');
hold off
drawnow
```

Fragmento de código 3.25. Diagrama de Bode y márgenes de fase y ganancia del sistema a controlar en PMGUIdiscbig

En cuanto a las funciones auxiliares, las funciones PMdiagrfunction, zdomain y fldrect se mantienen iguales en ambas herramientas cambiando la variable s por z, con cambios menores incluido que la función zdomain pasa a llamarse sdomain. La función colscheme se mantiene idéntica, mientras que la función NewCallback es sustituida por NewCallback_discr, en el fragmento de código 3.26, que en lugar de mostrar en el cuadro de texto interactivo comentado en el apartado 3.1 sobre la herramienta para sistemas continuos los valores de la magnitud y fase del punto del diagrama se muestran los valores de Z y del ángulo, valores propios en los sistemas discretos, además de cambiar la función por defecto para la que el valor mostrado es válido por la función por defecto incluida en el cuadro edit1 de la herramienta.

```
function output_txt = myfunction(obj,event_obj)
% Display the position of the data cursor
% obj          Currently not used (empty)
% event_obj    Handle to event object
% output_txt   Data cursor text string (string or cell array of strings).

pos = get(event_obj,'Position');
x=pos(1);
y=pos(2);
z=x+y*1j;
f=(0.1.*z)./((z-0.8).*(z-0.9));
ang=360-((angle(-f)+pi)*180/pi);

output_txt = {['X: ',num2str(pos(1),4)],...
    ['Y: ',num2str(pos(2),4)]};

% If there is a Z-coordinate in the position, display it as well
if length(pos) > 2
    output_txt{end+1} = ['Z: ',num2str(pos(3)+30,4)];
    output_txt{end+1} = ['Ang:',num2str(ang,4)];
end
```

Fragmento de código 3.26. Función auxiliar NewCallback_discr

Cálculo de controladores

La parte de código común a la que se hacía referencia en el punto 3.1 al explicar el cálculo de los diferentes controladores en la herramienta para sistemas continuos es muy similar en la aplicable a sistemas discretos. Los cambios que sufre, al igual que los explicados en la parte anterior a esta referente a la inicialización, representación del sistema y funciones auxiliares de la herramienta para sistemas discretos, están en su mayoría relacionados con cambios de la variable s a z y cambios de nombre en la consola gráfica, además de a la inclusión del cuadro de texto editable edit3 para la introducción de la posición de los polos deseados.

En cuanto al fragmento de código en el que se realiza la inicialización del cálculo, para la herramienta para sistemas continuos en el fragmento de código 3.9, se introduce un

fragmento en el que la aplicación comprueba si se ha introducido un valor de posición para los polos deseados, almacenando dicho valor en caso afirmativo y permitiendo al usuario introducir dicha posición seleccionando con el cursor en el diagrama en caso negativo. Además debido al cambio de s a z las variables sfun y zfun intercambian su nombre y la variable local newsfun pasa a llamarse newzfun. Por último se debe mencionar que no se realiza el cálculo de la posición de los polos característicos del sistema a controlar, debido a que el cálculo de controladores se realizará en esta herramienta utilizando únicamente los criterios del módulo y el argumento, cálculo que será explicado posteriormente, sin necesidad de utilizar los polos característicos en bucle cerrado como sí ocurría en la herramienta para sistemas continuos. Este fragmento se encuentra en el fragmento de código 3.27.

```
global sys sys2 controller

StringAxis=get(handles.edit2,'String');
ejes=str2num(StringAxis);
hax1=handles.axes1;
StringPoles=get(handles.edit3,'String');
NoPoles='Empty (Selected with cursor)';
if strcmp(StringPoles,NoPoles)==1
    DesiredPoles=0;
else
    DesiredPoles=str2num(StringPoles);
end

zfun=get(handles.edit1,'String');
sfun = strrep(zfun, 'z', 's');

zfun = strrep(zfun, './', '/');
newzfun=strrep(zfun,'.*','*');
z=tf('z');
sys=eval(newzfun);
if DesiredPoles==0
    [x,y]=ginput(1);
    plot(hax1,x,y,'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerSi
ze',7)
    plot(hax1,x,-y,'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerSi
ze',7)
    drawnow
else
    x=real(DesiredPoles); y=imag(DesiredPoles);
    plot(hax1,x,y,'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerSi
ze',7)
    plot(hax1,x,-y,'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerSi
ze',7)
    drawnow
end
```

Fragmento de código 3.27. Inicialización del cálculo de controladores en PMGUIdiscbig

En el fragmento en el que se representa el diagrama PM del sistema controlado, para la herramienta para sistemas continuos en el fragmento de código 3.10, las referencias al sistema de ejes axes2, en concreto las variables hax2 y h2, pasan a llamarse hax3 y h3 en referencia al sistema de ejes axes3, mientras que las variables newsfun y newzfun intercambian su nombre y newsfun2 pasa a llamarse newzfun2. Este fragmento se encuentra en el fragmento de código 3.28.

```
hax3=handles.axes3;
cla(hax3,'reset');
hold(hax3,'off');
axis(hax3,'off')
axis(hax3,'on')
PP=PMdiagrFunction(hax3,newsfun,ejes);%with controller1
newzfun2 = strrep(newzfun2, './', '/');newzfun2 = strrep(newzfun2, '.*',
'*');
sys2=eval(newzfun2);%with controller1
hold(hax3,'on');
h3=rlocusplot(hax3,sys2);%with controller1
p3=getoptions(h3);
p3.Title.String='';
p3.Xlabel.String=strcat('                ');
p3.XLabel.Color='w';
setoptions(h3,p3);
axis(ejes);
sgrid;
drawnow
hold(hax3, 'on')
%plot(hax3,x,y,'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerSiz
e',7)
%plot(hax3,x,-y,'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerSiz
e',7)
grid on
grid off
sgrid
box(hax3,'on')
drawnow
r2=rlocus(sys2);%with controller1
ax = axis(hax3);
hold on
plot(hax3,real(r2),imag(r2),'b.')
axis(ax);
drawnow
```

Fragmento de código 3.28. Diagrama PM del sistema controlado en PMGUIdiscbig.

Por último en lo referente a la parte común del cálculo, el código con el que se muestran en la consola gráfica la función de transferencia del controlador calculado y el diagrama de Bode y los márgenes de fase y ganancia del sistema controlado, en la herramienta para sistemas continuos en el fragmento de código 3.11, solo sufre cambios debido al cambio de nombres en la consola gráfica, de esta forma los cuadros de texto text5 y text2 pasan a llamarse text4

y text6 respectivamente, y el sistema de ejes hax3 pasa a llamarse hax2. Este fragmento se encuentra en el fragmento de código 3.29.

```
hax4=handles.axes4;
axis(hax4,'off')
axis(hax4,'on')
h4=bodeplot(hax4,Kc*sys2);
[Gm2,Pm2]=margin(Kc*sys2);
set(handles.text4,'String',strcat('Pm=',num2str(Pm2),',
Gm=',num2str(Gm2)));
hold on
margin(Kc*sys2);
%title('');
xlabel('');
hold off

Controller = strrep(Controller, './', '/');Controller = strrep(Controller,
'.*', '*');
set(handles.text6,'String',strcat('Controller=
',num2str(Kc),Controller));
controller=eval(Controller);
sys2=Kc*controller*sys;

pc2r=pole(feedback(sys2,1));
%hold(hax2, 'on')
for ii=1:length(pc2r)
    xx=real(pc2r(ii));yy=imag(pc2r(ii));
    if ax(1)<xx && xx<ax(2) && ax(3)<yy && yy<ax(4)
        plot(hax3,xx,yy,'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSiz
e',7)
    end
end
```

Fragmento de código 3.29. Controlador calculado, diagrama de Bode y márgenes de fase y ganancia del sistema controlado en PMGUIdiscbig

El cálculo de cada uno de los cuatro controladores básicos se realiza en dos pasos, en primer lugar se aplica el criterio del argumento para obtener la posición b del cero que incluye dicho controlador en el sistema, para luego aplicando el criterio del módulo teniendo en cuenta los polos y ceros introducidos por el controlador en el sistema calcular el valor de la ganancia Kc del controlador.

El criterio del argumento se aplica a partir de la ecuación 3.18, que indica que la diferencia entre los sumatorios de los ángulos que forman los polos del sistema con los polos deseados y los que forman los ceros es igual a 180 grados, y en la que α_i representa cada uno de los ángulos formados por polos del sistema y β_i cada uno de los ángulos formados por ceros del sistema, mientras que n_p y n_c representan respectivamente el número total de polos y ceros del sistema. Para el cálculo de cada uno de estos ángulos se utiliza o bien la ecuación 3.19 en caso de que la posición en el eje real del polo o cero del que se quiere calcular el ángulo se encuentre más a la izquierda que la de los polos deseados, o bien la ecuación 3.20 en el caso contrario. En dichas ecuaciones pc_i representa el polo o cero del que se está calculado el

ángulo, arctan la función arcotangente, p_d la posición de los polos deseados, siendo separada en parte real con real(p_d) e imaginaria con imag(p_d), y $\alpha\beta_i$ representa el ángulo calculado. Una vez calculados los ángulos y habiendo aplicado la ecuación 3.18 se obtiene el ángulo que forma con los polos deseados el cero que ha introducido el controlador en el sistema, y aplicando finalmente la ecuación 3.21, que no es más que la ecuación 3.19 particularizada y despejada para la posición b del cero que se desea calcular, y en la que β_b es el ángulo que forma dicho cero con los polos deseados. Este cálculo se encuentra en el fragmento de código 3.30.

$$\sum_{i=1}^{n_p} \alpha_i - \sum_{i=1}^{n_c} \beta_i = 180° \tag{3.18}$$

$$\alpha\beta_i = arctan\left(\frac{imag(p_d)}{real(p_d) - pc_i}\right) \tag{3.19}$$

$$\alpha\beta_i = 180° - arctan\left(\frac{imag(p_d)}{pc_i - real(p_d)}\right) \tag{3.20}$$

$$b = real(p_d) - \frac{imag(p_d)}{tan(\beta_b)} \tag{3.21}$$

Por otro lado el criterio del módulo se aplica a partir de la ecuación 3.22, que indica que la ganancia Kc del sistema es igual a la división entre el productorio de las distancias entre cada polo del sistema y la posición de los polos deseados, y el de las distancias entre cada cero del sistema y la posición de los polos deseados multiplicado por la ganancia del sistema sin controlar representada por K, y en la que d_i representa la distancia a los polos deseados de un polo o un cero y n_p y n_c representan respectivamente el número total de polos y ceros del sistema. Para el cálculo de estas distancias se utiliza o bien la ecuación 3.22 en caso de que la posición en el eje real del polo o cero del que se quiere calcular la distancia se encuentre más a la izquierda que la de los polos deseados, o bien la ecuación 3.23 en el caso contrario. En dichas ecuaciones pc_i representa el polo o cero del que se está calculado el ángulo y p_d la posición de los polos deseados, siendo separada en parte real con real(p_d) e imaginaria con imag(p_d). Este cálculo se encuentra en el fragmento de código 3.31.

```
poles=pole(sys);
[zeros,gain]=zero(sys);

for ii=1:length(poles)
    if poles(ii)<x
        alpha(ii)=atan(y/(x-poles(ii)))*(180/pi);
    else
        alpha(ii)=180-atan(y/(poles(ii)-x))*(180/pi);
    end
end
for ii=1:length(zeros)
    if zeros(ii)<x
        beta(ii)=atan(y/(x-zeros(ii)))*(180/pi);
    else
        beta(ii)=180-atan(y/(zeros(ii)-x))*(180/pi);
    end
end
betab=0;

for ii=1:length(poles)
    betab=betab+alpha(ii);
end
for ii=1:length(zeros)
    betab=betab-beta(ii);
end
betab=betab-180;
b=x-y/tan(betab*(pi/180));
```

Fragmento de código 3.30. Criterio del argumento en PMGUIdiscbig

$$K_c = \frac{\prod_{i=1}^{n_p} d_i}{K \cdot \prod_{i=1}^{n_c} d_i} \tag{3.22}$$

$$d_i = \sqrt{imag(p_d)^2 + (real(p_d) - pc_i)^2} \tag{3.23}$$

$$d_i = \sqrt{imag(p_d)^2 + (pc_i - real(p_d))^2} \tag{3.24}$$

```
for ii=1:length(poles)
    if poles(ii)<x
        d(ii)=sqrt((y^2)+((x-poles(ii))^2));
    else
        d(ii)=sqrt((y^2)+((poles(ii)-x)^2));
    end
end
for ii=1:length(zeros)
    if zeros(ii)<x
        d2(ii)=sqrt((y^2)+((x-zeros(ii))^2));
    else
        d2(ii)=sqrt((y^2)+((zeros(ii)-x)^2));
    end
end

dp=1;
for ii=1:length(poles)
    dp=dp*d(ii);
end
dz=1;
for ii=1:length(zeros)
    dz=dz*d2(ii);
end
Kc=dp/(gain*dz);
```

Fragmento de código 3.31. Criterio del módulo en PMGUIdiscbig

Para ejemplificar estos cálculos se introduce en la herramienta la ecuación 3.17 como función de transferencia del sistema a controlar, seleccionando como posición de los polos deseados elegida por el usuario 0.5 ± 0.5i introduciéndola directamente el cuadro de texto edit3. El diagrama PM de dicho sistema sin controlar con la posición de polos deseados indicada señalada en verde se encuentra en la figura 3.8.

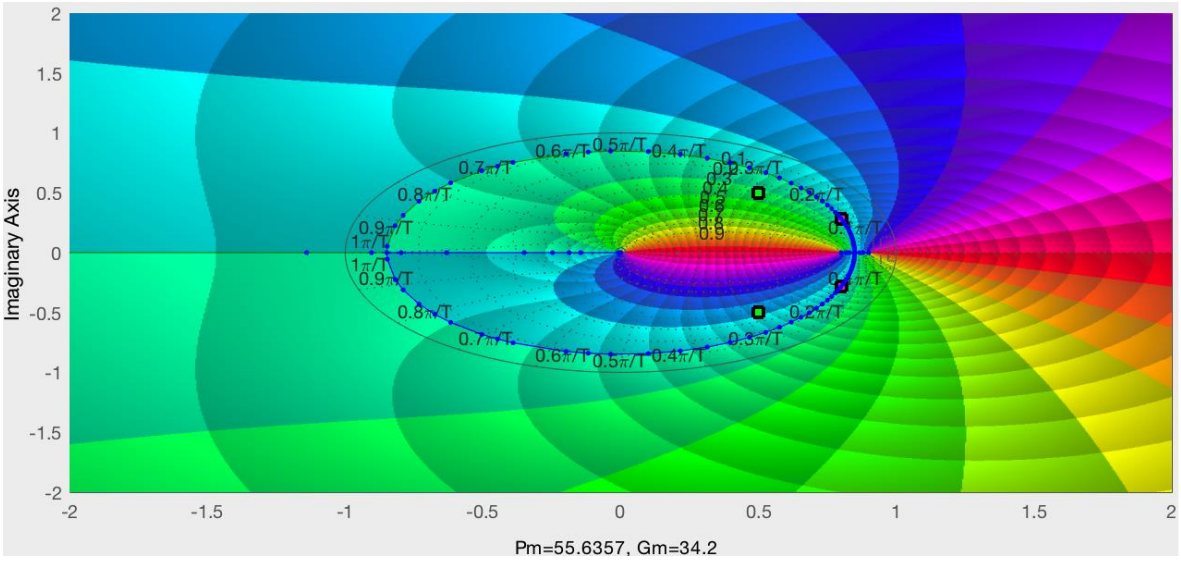

Figura 3.8. Diagrama PM de la función de transferencia en la ecuación 3.17 sin controlar

Para la aplicación de los criterios del módulo y el argumento explicados anteriormente es necesario reconocer los polos y ceros del sistema y analizar su posición con respecto a la de los polos deseados, para lo que se utiliza un diagrama representativo de los ejes real e imaginario como el de la figura 3.9, que representa los polos y ceros del sistema sin controlar con relación a la posición de los polos deseados. En dicho diagrama se utilizan las variables explicadas en las ecuaciones de los criterios del módulo y el argumento, siendo $alpha_i$ el ángulo que forma el polo con posición en i con los polos deseados, $beta_i$ el ángulo que forma el cero con posición en i con los polos deseados, d_i la distancia entre el polo o cero i y los polos deseados y p_d la posición de los polos deseados, en este caso en 0.5 ± 0.5i.

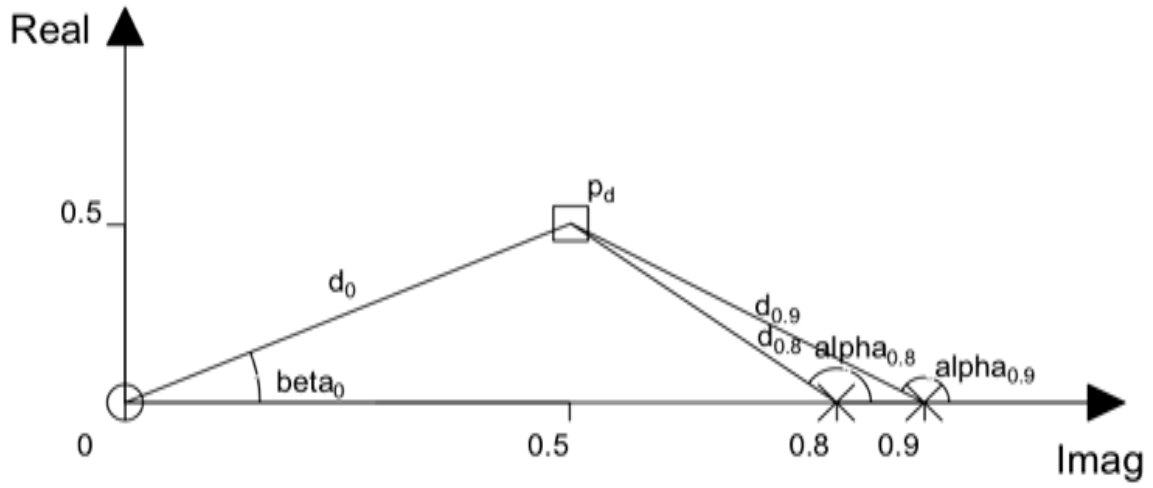

Figura 3.9. Diagrama para criterios del módulo y el argumento de la función de transferencia en la ecuación 3.17 sin controlar

La explicación del cálculo llevado a cabo por cada controlador se realizará utilizando el diagrama en la figura 3.9 incluyendo en él los polos o ceros que cada tipo de controlador introduzca en el sistema:

- Controlador P:

El controlador proporcional no introduce ningún polo o cero en el sistema, por lo que en su cálculo no se incluye el criterio del argumento, sino que se aplica directamente sobre el sistema a controlar el criterio del módulo en el fragmento de código 3.31 para calcular el valor de la ganancia Kc que será la función de transferencia del controlador tal como aparece en la ecuación 3.25.

$$controller = Kc \tag{3.25}$$

Para la función de transferencia elegida para ejemplificar el cálculo, el diagrama para aplicar el criterio del módulo es el mismo de la figura 3.9 al no introducir el controlador P ningún

polo o cero en el sistema, por lo que habrá tres distancias a calcular para aplicar la ecuación 3.22, cuyo cálculo se encuentra en las ecuaciones 3.26, 3.27 y 3.28 en las que se particulariza la ecuación 3.23 para el cálculo de la distancia del cero situado en 0 al estar a la izquierda de la posición de los polos deseados, y la ecuación 3.24 para los polos situados en 0.8 y 0.9 al estar a la derecha de dicha posición.

$$d_0 = \sqrt{0.5^2 + (0.5 - 0)^2} = 0.7071 \tag{3.26}$$

$$d_{0.8} = \sqrt{0.5^2 + (0.8 - 0.5)^2} = 0.5831 \tag{3.27}$$

$$d_{0.9} = \sqrt{0.5^2 + (0.9 - 0.5)^2} = 0.6403 \tag{3.28}$$

La particularización de la ecuación 3.22 para este sistema se encuentra en la ecuación 3.29, en la que se calcula el valor de la ganancia Kc, resultando por lo tanto al aplicar la ecuación 3.25 en el controlador P de la ecuación 3.30. En la figura 3.10 está representado el sistema controlado por dicho controlador P, comprobándose que la posición de los polos característicos del sistema se encuentra más cerca de la posición deseada que antes de ser controlado, aunque sin llegar a alcanzarla debido a que la posición deseada no se encuentra sobre el lugar de las raíces del sistema sin controlar y por lo tanto es imposible llevar hasta dicha posición la de los polos característicos utilizando un controlador proporcional.

$$K_c = \frac{d_{0.8} d_{0.9}}{K d_0} = \frac{0.5831 \cdot 0.6403}{0.1 \cdot 0.7071} = 5.2802 \tag{3.29}$$

$$controller = 5.2802 \tag{3.30}$$

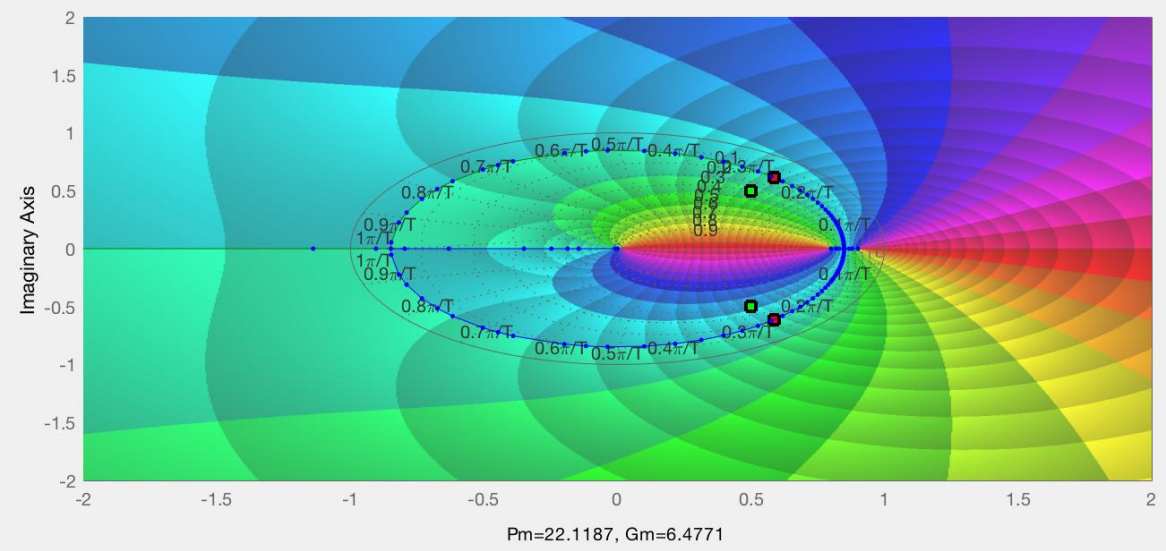

Figura 3.10. Diagrama PM de la función de transferencia en la ecuación 3.17 controlada con controlador P

- Controlador PI:

El controlador integral, a diferencia de los demás, no es utilizado con el objetivo de cambiar los polos característicos del sistema, sino de reducir el error en régimen estacionario del sistema manteniendo el comportamiento dinámico del mismo, por lo que en el código para el controlador PI se elimina la selección de polos deseados por el usuario tanto con el cursor como con el cuadro de texto edit3 y no se aplican los criterios del módulo y el argumento para realizar el cálculo. Para ejercer el control introduce en el sistema un polo en la posición 1 para eliminar de esta forma el error en régimen permanente del sistema, además de un cero en una posición muy cercana a este polo con el fin de compensar el polo introducido, en este caso en una posición ab calculada utilizando una aproximación práctica similar a la de la ecuación 3.4 para sistemas discretos, que sitúa dicho cero a una distancia de la posición 1 de un sexto de la distancia en el eje real entre los polos característicos del sistema en bucle cerrado antes de ser controlado y dicha posición 1, manteniendo la ganancia unitaria al igual que en el cálculo del controlador integral para sistemas continuos. En la ecuación 3.31 se encuentra la función de transferencia del controlador PI para sistemas discretos, encontrándose en la ecuación 3.32 el cálculo de la posición ab del cero introducido en el sistema por el controlador.

$$controller = Kc\frac{(z-ab)}{(z-1)} \tag{3.31}$$

$$ab = 1 - \frac{1-|real(pc)|}{6} \tag{3.32}$$

Antes del cálculo de la posición ab debe comprobarse si el sistema posee ya un polo en 1, en cuyo caso el controlador PI solo introduciría en el sistema el cero en ab, lo cual es realizado con la modificación del código presente en el fragmento de código 3.32.

```
pole1=0;
for i=1:length(poles)
    if poles(i)==1
        pole1=1;
    end
end
if pole1==0
poles(length(poles)+1)=1;
end
```

Fragmento de código 3.32. Comprobación de polo en 1 para controlador PI en PMGUIdiscbig

Una vez realizada la comprobación se calcula la posición ab del cero con el fragmento de código 3.33. Particularizando la ecuación 3.32 para el caso del ejemplo se obtiene la posición para el cero calculada en la ecuación 3.33, representándose además en la ecuación 3.34 la asignación del valor 1 como la ganancia del controlador,.

```
sysc=feedback(sys,1);
pc=pole(sysc);
ab=1-((1-abs(real(pc(2))))/6);
```

Fragmento de código 3.33. Cero del controlador PI en PMGUIdiscbig

$$ab = 1 - \frac{1 - |real(pc)|}{6} = 0.9667 \tag{3.33}$$

$$Kc = 1 \tag{3.34}$$

Con los resultados en la ecuaciones 3.33 y 3.34 se particulariza la ecuación 3.31 para este controlador, en la ecuación 3.35, estando representado en la figura 3.11 el diagrama PM del sistema controlado con dicho controlador, comprobándose que la variación sobre el comportamiento dinámico del sistema es mínima al introducir el polo y el cero en posiciones muy cercanas, consiguiendo anular el error en régimen estacionario del sistema al introducir un polo en 1, por lo que el control es correcto.

$$controller = \frac{(z - 0.9667)}{(z - 1)} \tag{3.35}$$

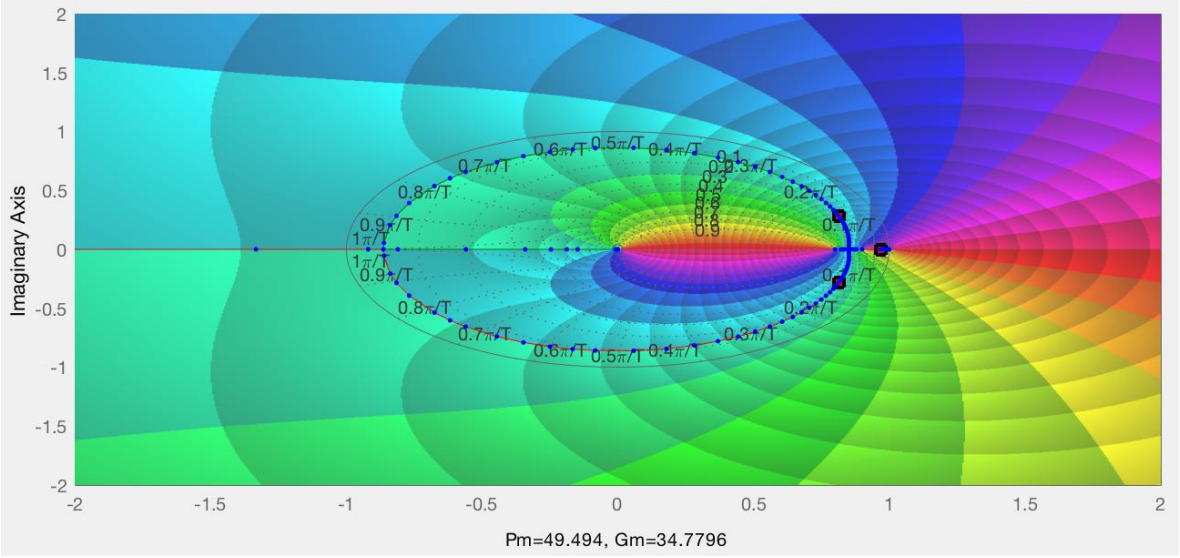

Figura 3.11. Diagrama PM de la función de transferencia en la ecuación 3.17 controlada con controlador PI

- Controlador PD real:

El controlador derivativo introduce un polo en 0 y un cero en una posición b calculada utilizando el criterio del argumento. La función de transferencia de un controlador PD real

en el plano discreto se encuentra en la ecuación 3.36 mientras que el diagrama del sistema para la aplicación de los criterios del módulo y el argumento se encuentra en la figura 3.12, incluyendo en el de la figura 3.9 un polo en 0 y un cero en b introducidos por el controlador PD.

$$controller = Kc\frac{(z-b)}{z} \tag{3.36}$$

Para obtener la posición b del cero que introduce el controlador en el sistema se aplica el criterio del argumento calculando los ángulos que forman cada polo y cero del sistema con la posición de los polos deseados. Observando el diagrama se aprecia que en la posición 0 existen dos ángulos de igual magnitud, α_0 y β_0, debido a que en la misma posición existen un cero y un polo. La ecuación 3.18 con la que se aplica el criterio del argumento postula que se debe sustraer al ángulo total formado por los polos el ángulo formado por los ceros, lo que hace que, al tener ambos ángulos formados desde 0 el mismo valor, no sea necesario incluir dichos ángulos en el cálculo ya que el resultado no varía al anularse entre ellos.

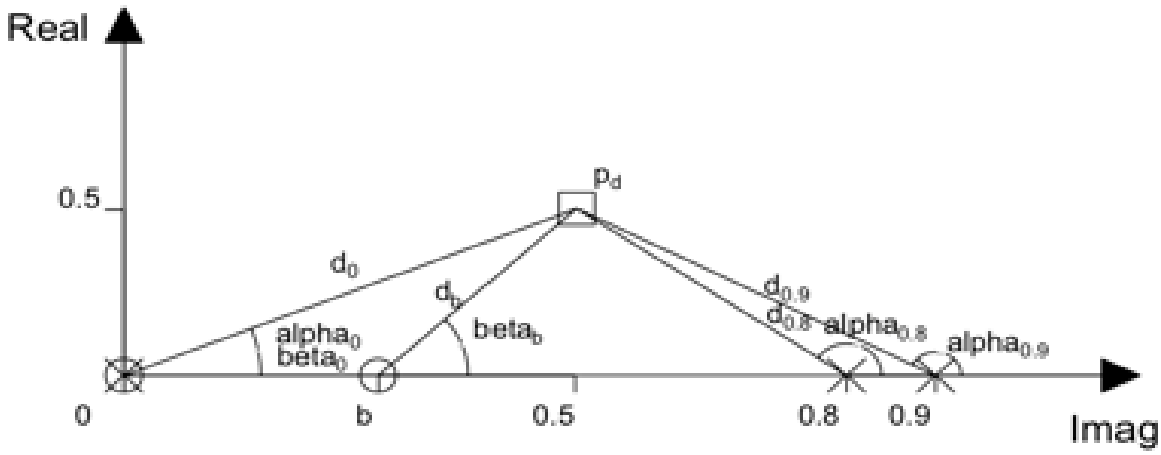

Figura 3.12. Diagrama para criterios del módulo y el argumento de la función de transferencia en la ecuación 3.17 controlada con controlador PD real

De esta forma solo son necesarios los valores de los ángulos en los polos en 0.8 y 0.9, calculados en las ecuaciones 3.37 y 3.38 al particularizar la ecuación 3.20, por lo que se puede particularizar directamente la ecuación 3.18 para obtener el ángulo formado por el cero en b, en la ecuación 3.39. Conociendo el ángulo se particulariza la ecuación 3.21 para obtener la posición del cero introducido, en la ecuación 3.40. La herramienta realiza este cálculo introduciendo tras el almacenamiento de la posición de los polos del sistema a controlar en el fragmento de código 3.30 para la aplicación del criterio del argumento, la línea en el fragmento de código 3.34 que añade al conjunto de polos del sistema el polo en 0.

$$\alpha_{0.8} = 180° - arctan\left(\frac{0.5}{0.8 - 0.5}\right) = 120.9638° \tag{3.37}$$

$$\alpha_{0.9} = 180° - arctan\left(\frac{0.5}{0.9 - 0.5}\right) = 128.6598° \tag{3.38}$$

$$\beta_b = \alpha_{0.8} + \alpha_{0.9} - 180° = 120.9638° + 128.6598° - 180° = 69.6236° \tag{3.39}$$

$$b = 0.5 - \frac{0.5}{tan(69.6236°)} = 0.3143 \tag{3.40}$$

```
poles(length(poles)+1)=0;
```

Fragmento de código 3.34. Modificación del criterio del argumento para controladores PD y PID real en PMGUIdiscbig

Una vez conocida la posición del cero se calcula el valor de la ganancia Kc aplicando el criterio del argumento, sin tener en cuenta la distancia desde la posición 0 debido a que al haber allí un polo y un cero sus distancias se anularían al particularizar la ecuación 3.22 al estar cada una en un lado de la división y tener la misma magnitud.

Esto hace que la particularización de la ecuación 3.22 solo incluya los polos en 0.8 y 0.9, cuya distancia es calculada en las ecuaciones 3.27 y 3.28 respectivamente, y el cero introducido en la posición b, cuya distancia se calcula en la ecuación 3.41 en la que se particulariza la ecuación 3.23 para el cero en b, dando como resultado la ganancia Kc en la ecuación 3.42.

$$d_b = \sqrt{0.5^2 + (0.5 - 0.3143)^2} = 0.5334 \tag{3.41}$$

$$K_c = \frac{d_{0.8} d_{0.9}}{K d_{0.3143}} = \frac{0.5831 \cdot 0.6403}{0.1 \cdot 0.5334} = 7 \tag{3.42}$$

Con los resultados de las ecuaciones 3.40 y 3.42 se puede particularizar la ecuación 3.36 para obtener la función de transferencia del controlador calculado en la ecuación 3.43. El diagrama PM del sistema controlado por el controlador PD calculado se encuentra en la figura 3.13, en la que se aprecia que los polos del sistema en bucle cerrado se encuentran en la posición de los polos deseados, por lo que el control es correcto.

$$controller = 7\,\frac{(z - 0.3143)}{z} \tag{3.43}$$

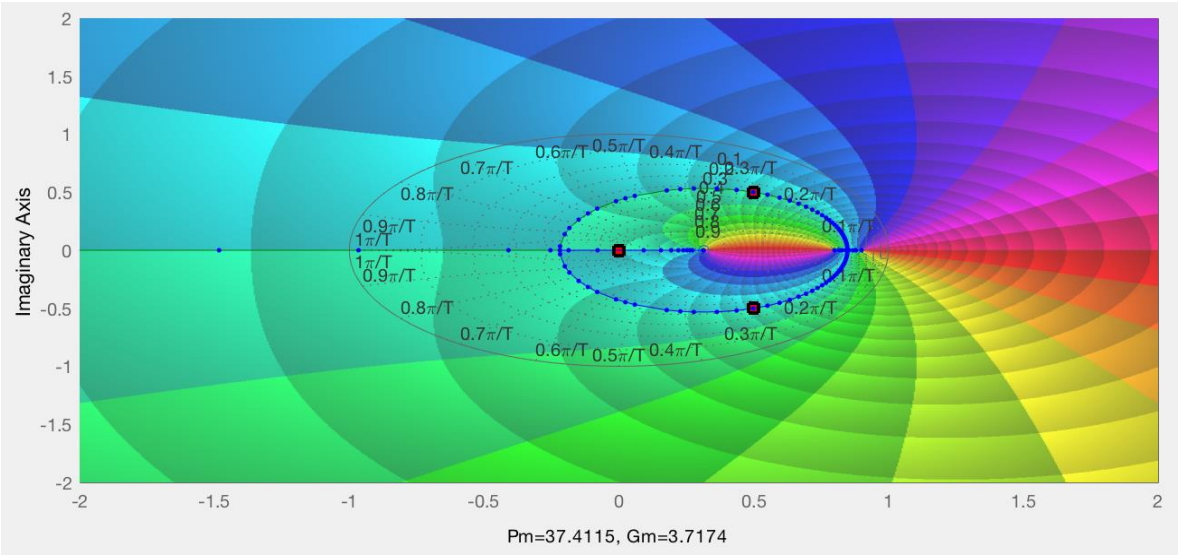

Figura 3.13. Diagrama PM de la función de transferencia en la ecuación 3.17 controlada con controlador PD real

- Controlador PID real:

El controlador integral-derivativo se calcula de manera muy similar al derivativo, ya que introduce un polo en 0 y un cero en una posición b calculada con el criterio del argumento al igual que el controlador PD, y además introduce un polo en 1 y un cero en una posición muy cercana a 1, lo que hace que al estar muy cerca del polo anule su efecto sobre el comportamiento transitorio del sistema pero haciendo el error en régimen permanente del sistema nulo al introducir un polo en 1, dando lugar a la función de transferencia en la ecuación 3.48.

$$controller = Kc\frac{(z-b)}{z}\frac{(z-0.99)}{(z-1)} \tag{3.44}$$

El cálculo se realiza en dos etapas como en los anteriores controladores, con la diferencia de que con el controlador PID se pretende solucionar tanto en comportamiento del sistema en régimen permanente como en transitorio. Para hacer el control en régimen transitorio más efectivo, el cálculo de la posición del cero que lo controla se realiza de manera idéntica a la del control PD explicado anteriormente ya que no se incluyen en él el cero en 0.99 y el polo en 1 ya que su cometido es eliminar el error en régimen permanente.

Esto provoca que las particularizaciones de las ecuaciones 3.18 y 3.21 sean iguales que las del controlador PD en las ecuaciones 3.39 y 3.40, dando como resultado la misma posición b para el cero.

En la aplicación del criterio del módulo sí se tienen en cuenta el polo y cero introducidos por el control integral, por lo que al particularizar la ecuación 3.22 se incluyen las distancias de

los polos en 0.8, 0.9 y 1 y los ceros en b y 0.99, dando lugar a la ecuación 3.45. Las distancias calculadas están en las ecuaciones 3.27, 3.28, 3.32, 3.33 y 3.41.

$$K_c = \frac{d_{0.8} d_{0.9} d_1}{K d_{0.3143} d_{0.99}} = \frac{0.5831 \cdot 0.6403 \cdot 0.7071}{0.1 \cdot 0.5334 \cdot 0.7001} = 7.07 \tag{3.45}$$

Con los resultados de las ecuaciones 3.40 y 3.45 se particulariza la ecuación 3.44 para este caso dando lugar al controlador en la ecuación 3.46. El diagrama PM del sistema controlado por el controlador PID calculado se encuentra en la figura 3.14, en la que se aprecia un sistema controlado de manera muy similar al de la figura 3.13, pero con un polo en 1 lo que provoca que el error en régimen permanente sea nulo, además de que el control sea correcto.

$$controller = 7.07 \frac{(z - 0.3143)}{z} \frac{(z - 0.99)}{(z - 1)} \tag{3.46}$$

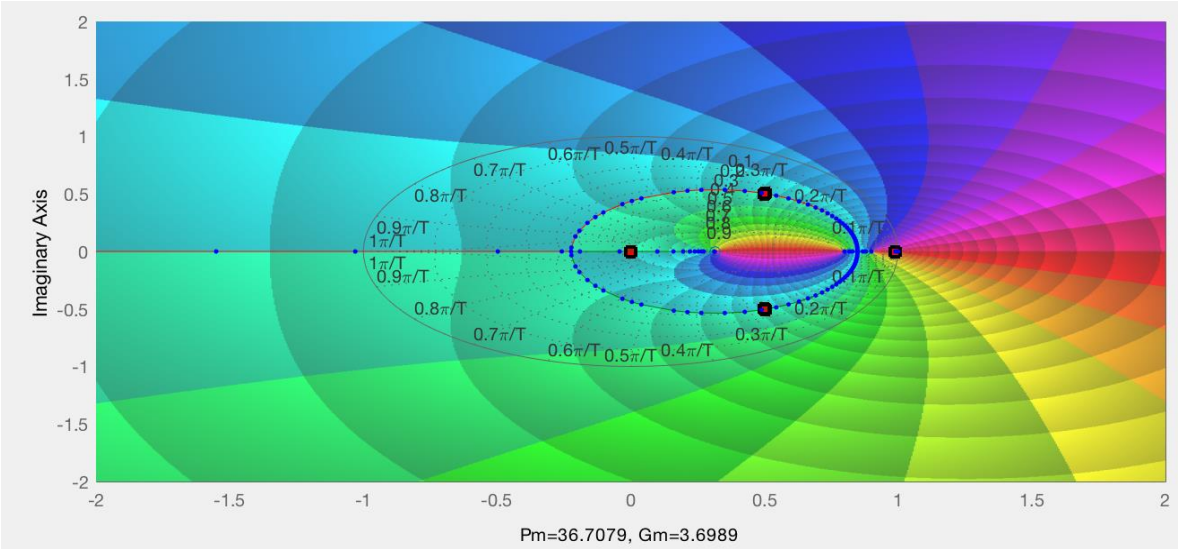

Figura 3.14. Diagrama PM de la función de transferencia en la ecuación 3.17 controlada con controlador PID real

Opciones de análisis extra

El código referente a las cinco opciones de análisis presentes en la herramienta se mantienen idénticos a los de la herramienta para sistemas continuos, por lo que carece de interés su explicación de nuevo en este apartado.

```
%% Reset Transfer Function
% --- Executes on button press in pushbutton14.
function pushbutton14_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Reset='(0.1.*z)./((z-0.8).*(z-0.9))';
set(handles.edit1,'String',Reset);
```

Fragmento de código 3.35. Opción Reset Transfer Function en PMGUIdiscbig

En el fragmento de código 3.35 se encuentra el código activado al pulsar el botón Reset del cuadro de texto edit1 en el que se introduce la función de transferencia del sistema a controlar, que restaura su valor por defecto. El código que realiza el Reset para los cuadros de texto edti2 y edit3, en los que se introducen los límites de los sistemas de ejes en los que se representan los diagramas PM y la posición de los polos deseados por el usuario respectivamente, sigue la misma estructura, sustituyendo el valor de la variable Reset por los valores por defecto de cada cuadro de texto, y cambiando la variable handles.edit1 por la referente a su cuadro de texto correspondiente.

3.4. Inclusión de controladores Lead, Lag y Lead-Lag Network

La teoría para el desarrollo en Matlab de redes o controladores de adelanto, atraso y adelanto-atraso (lead, lag y lead-lag) ha sido extraída del libro sobre resolución de problemas de control en el plano discreto utilizando Matlab. [21]

La decisión de incluir este tipo de controladores en la herramienta surge debido a que el método del lugar de las raíces y los criterios del módulo y el argumento es válido cuando los objetivos de diseño son especificaciones como la sobreoscilación o el número de muestras de subida o estabilización, ya que estas modifican la posición de los polos característicos del sistema y son por tanto controlables aplicando las técnicas antes mencionadas. Sin embargo, cuando los objetivos de diseño incluyen especificaciones relacionadas con la robustez, como los márgenes de fase y ganancia, el diseño del controlador debe realizarse en el dominio de la frecuencia en lugar de en el del tiempo, por lo que no son aplicables los mismos métodos que para los controladores del código básico.

Puesto que el diseño se realiza en el dominio de la frecuencia, resulta más sencillo trabajar con funciones de transferencia en el plano continuo en lugar de en el discreto, por lo que previamente al cálculo de cada uno de los controladores la función de transferencia del sistema a controlar será transformada al plano continuo usando la transformada bilineal o método de Tustin, en la ecuación 3.47, siendo finalmente la función de transferencia del

controlador transformada de nuevo al plano discreto utilizando la transformada bilineal inversa, en la ecuación 3.48.

La elección de la transformada bilineal en lugar de otras para convertir funciones de transferencia del plano continuo de la variable s al plano discreto de la variable z está razonada debido a que esta convierte el eje imaginario del plano s en el círculo unitario del plano z, por lo que mantiene las propiedades de respuesta en frecuencia del sistema de control realimentado al corresponder cada frecuencia en el plano continuo a una frecuencia en el plano discreto siguiendo la ecuación 3.49, en la que w_s representa la frecuencia en el plano continuo y w_z la frecuencia en el plano discreto. En las ecuaciones 3.47, 3.48 y 3.49 la variable T_s representa el tiempo de muestreo del sistema.

$$s = \frac{2}{T_s}\frac{(z-1)}{(z+1)} \tag{3.47}$$

$$z = \frac{1 + \frac{sT_s}{2}}{1 - \frac{sT_s}{2}} \tag{3.48}$$

$$w_s = \frac{2}{T_s}\tan\left(\frac{w_z T_s}{2}\right) \tag{3.49}$$

Para la ejemplificación de cada uno de los tres controladores se utilizará la función de transferencia en la ecuación 3.50 como función de transferencia del sistema a controlar ya que la de la ecuación 3.17 utilizada anteriormente ya es un sistema estable con un aceptable margen de fase antes de ser controlada, al contrario que la de la ecuación 3.50. En este caso al estar las especificaciones relacionadas con la robustez del sistema no se realiza selección de polos deseados por el usuario, sino que se escogerán valores para el error en régimen estacionario y el margen de fase del sistema.

La función de transferencia en la ecuación 3.50 sin controlar se encuentra representada en el diagrama PM en la figura 3.15, en la que puede comprobarse que el sistema es inestable en lazo cerrado debido a que sus polos característicos se encuentran en 0.35 ± 1.26i, fuera del círculo unitario, por lo que el valor de su margen de fase es de -30.6º además de tener un error en régimen estacionario de 0.0099 y el efecto de los controladores será claramente apreciable.

$$G(z) = \frac{(z+1)}{(z-0.8)(z-0.9)} \tag{3.50}$$

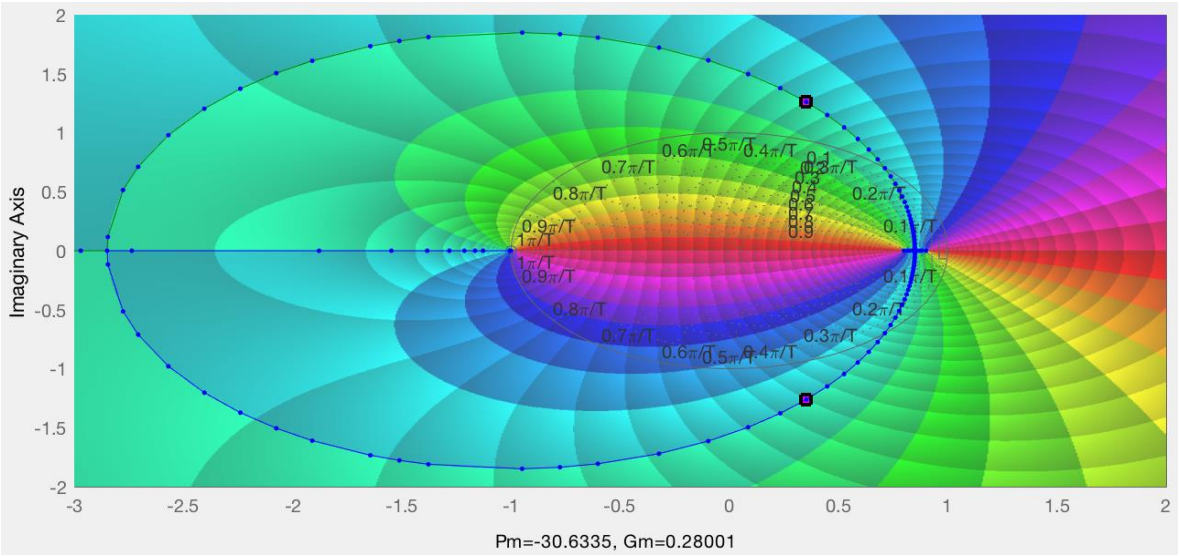

Figura 3.15. Diagrama PM de la función de transferencia en la ecuación 3.50 sin controlar

En cuanto al código, los tres controladores compartirán una parte común relativa a la inicialización, la representación del diagrama PM del sistema controlado y la representación del controlador calculado y del diagrama de Bode del sistema controlado, al igual que ocurría en el código de los cuatro controladores básicos, con alguna diferencia.

El código relacionado con la inicialización es muy similar al código en el fragmento de código 3.27, eliminando lo referente al cuadro de texto edit3 y a la representación de los polos deseados al no haberlo, y cambiando la línea en la que se define z por la línea en el fragmento de código 3.36 en la que además de definir z como la variable con la que operar se establece también el tiempo de muestreo del sistema en 0.1 s, debido a que para la transformación al plano continuo es necesario un tiempo de muestreo.

```
z=tf('z',0.1);
```

Fragmento de código 3.36. Modificación de la inicialización para redes en PMGUIdiscbig

El código para la representación del diagrama PM del sistema controlado no varía respecto al fragmento de código 3.28, mientras que en lo referente a la representación del controlador calculado y del diagrama de Bode del sistema controlado en el fragmento de código 3.29 la única variación es que la línea en la que se almacenan los valores de los márgenes de ganancia y fase del sistema controlado es cambiada por la línea en el fragmento de código 3.37, que además de almacenar los valores de los márgenes también almacena en que frecuencia se alcanzan cada uno de esos márgenes.

```
[Gm2,Pm2,Wgm,Wpm]=margin(sys2);
```

Fragmento de código 3.37. Modificación del cálculo de márgenes de ganancia y fase del sistema controlado para redes en PMGUIdiscbig

Controlador Lead (Lead Network)

El controlador o red de adelanto de fase ejerce una función similar a la del controlador PD en el dominio del tiempo, pero actuando en el dominio de la frecuencia, ya que reduce el tiempo de subida del sistema y aumenta el ancho de banda de la respuesta en frecuencia del sistema realimentado en bucle cerrado, incrementando la magnitud de las frecuencias medias y altas y adelantando la fase.

La función de transferencia de un controlador Lead en el plano continuo se encuentra en la ecuación 3.51, por lo que se deberán calcular tres parámetros: la ganancia de adelanto del sistema K_{lead}, la posición del cero que introduce en el sistema z_{lead} y un coeficiente α_{lead} relacionado con el margen de fase del sistema.

$$controller = K_{lead}\alpha_{lead}\frac{s - z_{lead}}{s - \alpha_{lead}z_{lead}} \tag{3.51}$$

Para aplicar el método propuesto en la fuente [21] es necesario fijar unos valores deseados de margen de fase y de error en régimen estacionario. Se han fijado uno valores de 70º como margen de fase mínimo y 0.1 como error en régimen estacionario máximo debido a que con tales valores se alcanza un sistema razonablemente robusto, siempre siendo posible cambiar estos valores en función de los requisitos del sistema. Además, se transforma la función de transferencia del sistema a controlar al plano s utilizando la transformada bilineal, resultando la función de transferencia en s representada con la variable local syss.

Tras esta inicialización se calcula la ganancia de adelanto del sistema, para lo que se requiere primero la constante de posición del sistema, o ganancia de lazo abierto en baja frecuencia, calculada a partir del error en régimen estacionario máximo fijado con la ecuación 3.52 en la que e_{ss} representa el error en régimen estacionario del sistema, puesto que el sistema es de tipo 0 al no tener polos en s=0; y la ganancia DC de la planta, o lo que es lo mismo el valor de la función de transferencia del sistema a controlar cuando s vale 0, calculada en la ecuación 3.53 en la que G(s) representa la función de transferencia del sistema a controlar transformada al plano continuo. Finalmente, con la ecuación 3.54 se obtiene el valor de la ganancia de adelanto.

$$K_p = \frac{1}{e_{ss}} - 1 \tag{3.52}$$

$$plant_{lfg} = G(s = 0) \tag{3.53}$$

$$K_{lead} = \frac{K_p}{plant_{lfg}} \tag{3.54}$$

Para el cálculo del valor del coeficiente y la posición del cero se realiza un estudio del sistema en frecuencia. Debido a que la curva entra magnitud y frecuencia tiene pendiente negativa, el valor de margen de fase que se debe buscar en el cálculo debe ser ligeramente mayor que el fijado, ya que tenderá a reducirse debido a dicha pendiente, lo que en base a la experiencia se traduce en un incremento de 7º, un 10 % sobre el valor de margen de fase mínimo fijado.

Para calcular la magnitud del incremento en el margen de fase es necesario conocer algún valor de la fase del sistema, lo cual es estudiado en una frecuencia central prefijada en 5 rad/s, ya que es un valor suficiente para acelerar la respuesta del sistema en el dominio del tiempo. Conociendo este valor se aplica la ecuación 3.55 para calcular la variación que debe producirse en el margen de fase, en la que Δ_{PM} representa dicha variación, ph representa la fase medida en la frecuencia central, y PM_{target} representa el valor objetivo del margen de fase sumando el incremento del 10 % antes mencionado.

$$\Delta_{PM} = PM_{target} - (180° + ph) \tag{3.55}$$

Finalmente, con la ecuación 3.56 se calcula el valor del coeficiente α_{lead} y con la ecuación 3.57 la posición del cero z_{lead}, en la que ww representa el valor de la frecuencia central seleccionada anteriormente, en este caso 5 rad/s. El código referente al cálculo de los parámetros del controlador Lead se encuentra en el fragmento de código 3.38.

$$\alpha_{lead} = \frac{1 + sen(\Delta_{PM})}{1 - sen(\Delta_{PM})} \tag{3.56}$$

$$z_{lead} = \frac{-ww}{\sqrt{\alpha_{lead}}} \tag{3.57}$$

```
%Data

PM=70;
ess=0.1;
Ts=0.1;

syss=d2c(sys,'tustin');

%Lead gain

Kp_read=(1/ess)-1;
plant_lfg=dcgain(syss);
Klead=Kp_read/plant_lfg;

%Lead alpha and zero

ww=5;
pm_target=PM+7;
[mag,ph]=bode(Klead*syss,ww);
del_pm=pm_target-(180+ph);
alphalead=(1+sin(del_pm*pi/180))/(1-sin(del_pm*pi/180));
zlead=-ww/sqrt(alphalead);
```

Fragmento de código 3.38. Controlador Lead Network en PMGUIdiscbig

Para ejemplificar el cálculo se utiliza la función de transferencia antes mencionada en la ecuación 3.50, cuya función de transferencia al transformarla al plano continuo se encuentra en la ecuación 3.58, introduciendo como valores de margen de fase mínimo y error en régimen estacionario máximo los establecidos anteriormente. De esta manera se particularizan las ecuaciones 3.52, 3.53, 3.54, 3.55, 3.56 y 3.57 en las ecuaciones 3.59, 3.60, 3.61, 3.62, 3.63 y 3.64 respectivamente.

$$G(s) = \frac{-11.7s + 233.9}{s^2 + 3.275s + 2.339} \tag{3.58}$$

$$K_p = \frac{1}{0.1} - 1 = 9 \tag{3.59}$$

$$plant_{lfg} = \frac{-11.7 \cdot 0 + 233.9}{0^2 + 3.275 \cdot 0 + 2.339} = 100 \tag{3.60}$$

$$K_{lead} = \frac{9}{100} = 0.09 \tag{3.61}$$

$$\Delta_{PM} = 77° - (180° + 201.8149°) = -304.8149° \tag{3.62}$$

$$\alpha_{lead} = \frac{1 + sen(-304.8149°)}{1 - sen(-304.8149°)} = 10.1732 \tag{3.63}$$

$$z_{lead} = \frac{-5}{\sqrt{10.1732}} = -1.5676 \tag{3.64}$$

Con los resultados de las ecuaciones 3.61, 3.63 y 3.64 puede particularizarse para este caso la ecuación 3.51 en la ecuación 3.65 del controlador Lead calculado en el plano continuo, que se transforma al plano discreto, en la ecuación 3.66, para obtener la función de transferencia del controlador en z. El diagrama PM del sistema controlado se muestra en la figura 3.16, en la que se comprueba que el sistema es ahora estable con un margen de fase de 28.5º, lejano al valor prefijado de 70º ya que se debía incrementar el valor del margen de fase en 100º, tarea complicada para un controlador Lead simple. Además, se debe mencionar que el error en régimen estacionario pasa de tener un valor de 0.0099 en el sistema sin controlar a 0.099 al actuar el controlador Lead, por lo que en términos generales el controlador consigue controlar el sistema, aunque sin alcanzar el valor requerido de margen de fase para constituir un sistema robusto.

$$controller = 0.09 \cdot 10.1732 \frac{s + 1.5676}{s - 10.1732 \cdot (-1.5676)} = 0.9156 \frac{s + 1.5676}{s + 15.9475} \tag{3.65}$$

$$controller = 0.5493 \frac{z - 0.8547}{z - 0.1127} \tag{3.66}$$

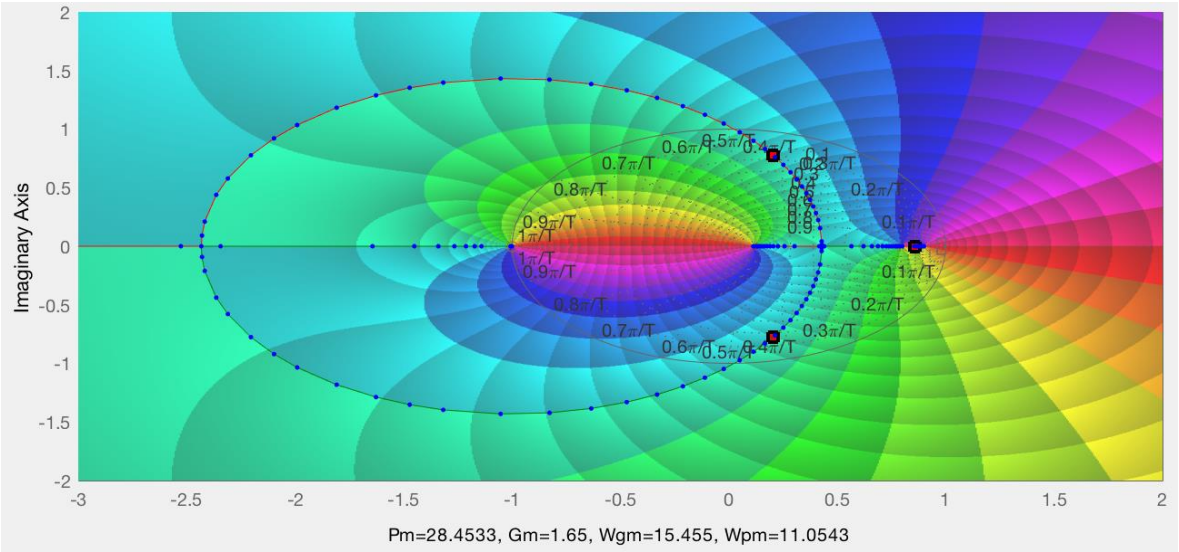

Figura 3.16. Diagrama PM de la función de transferencia en la ecuación 3.50 controlada con controlador Lead Network

Controlador Lag (Lag Network)

El controlador o red de atraso de fase ejerce una función similar a la del control PI, ya que la consecuencia de su control es la disminución del error en régimen estacionario y la sobreoscilación del sistema.

Para la variación del control el cálculo se inicia en torno a dos parámetros establecidos por el usuario, el margen de fase mínimo que debe tener el sistema y que afectará al comportamiento dinámico del mismo y por lo tanto a su sobreoscilación, y la ganancia de lazo abierto en baja frecuencia mínima, que afectará al error en régimen estacionario del sistema siguiendo la ecuación 3.52.

La función de transferencia de un controlador Lag en el plano continuo se encuentra en la ecuación 3.67, por lo que se deberán calcular tres parámetros: la ganancia de atraso del sistema K_{lag}, la posición del cero que introduce en el sistema z_{lag} y un coeficiente α_{lag} relacionado con el comportamiento en régimen estacionario del sistema.

$$controller = K_{lag}\frac{s - z_{lag}}{s - {z_{lag}}/{\alpha_{lag}}} \tag{3.67}$$

Los valores seleccionados para los parámetros del cálculo son 110º como margen de fase mínimo, ya que el control con redes de atraso no suele utilizarse sin compensarla con una red de adelanto y no alcanza márgenes de fase elevados con exactitud y 10 como ganancia de lazo abierto en baja frecuencia, lo que equivale a un error en régimen estacionario de 0.09 aproximadamente. Se realiza esta elección de valores debido a que se ha comprobado experimentalmente que con ellos se alcanza un sistema razonablemente robusto, siempre

siendo posible cambiar estos valores en función de los requisitos del sistema. Además, se transforma la función de transferencia del sistema a controlar al plano s utilizando la transformada bilineal, resultando la función de transferencia en s representada con la variable local syss.

Tras esta inicialización se calcula la ganancia de atraso del sistema, para lo que se genera un vector ww de 100 puntos espaciados logarítmicamente entre 0.1 y 10, para luego almacenar en los vectores mag_ratio y ph los valores de la magnitud en escala natural y de la fase del sistema respectivamente en los 100 puntos generados anteriormente. Utilizando el dato de margen de fase mínimo se aplica la ecuación 3.68 para obtener el parámetro β que será el margen de fase buscado del sistema, en la que PM representa el dato de margen de fase mínimo. Para encontrar la magnitud del sistema en la que su margen de fase es más cercano al parámetro β se realiza una interpolación lineal entre los vector ph y mag_ratio en el punto β, encontrando el valor mag_β que representa la magnitud del sistema buscada y con la que se obtiene la ganancia de atraso del sistema utilizando la ecuación 3.69.

$$\beta = (PM + 5°) - 180° \tag{3.68}$$

$$K_{lag} = \frac{1}{mag_\beta} \tag{3.69}$$

A continuación, se calculan los valores del coeficiente α_{lag} y del cero z_{lag}. El primero de ellos mantendrá la ganancia de lazo abierto en baja frecuencia del sistema en el valor seleccionado, para ello se calcula con la ecuación 3.70 la ganancia DC de la planta junto con la ganancia de atraso calculada, para, aplicando la ecuación 3.71 en la que K_{low} representa el valor seleccionado como ganancia de lazo abierto en baja frecuencia mínima deseado, obtener el valor del coeficiente. Para calcular la posición del cero se calcula un valor de frecuencia central ww_c como la interpolación entre los vectores ph y ww en el punto β, que será utilizada en la ecuación 3.72 para obtener la posición de un cero que mueve el margen de fase a un valor cercano al introducido como parámetro. El código referente al cálculo de los parámetros del controlador Lag se encuentra en el fragmento de código 3.39.

$$plant_{lfg} = K_{lag} G(s = 0) \tag{3.70}$$

$$\alpha_{lag} = \frac{K_{\mathrm{low}}}{plant_{\mathrm{lfg}}} \tag{3.71}$$

$$z_{lag} = -\left(\frac{ww_c}{10} - \frac{ww_c}{100}\right) \tag{3.72}$$

```
%Data

PM=110;
Klow=10;
Ts=0.1;
syss=d2c(sys,'tustin');
%Lag gain

ww=logspace(-1,1,100);
[mag_db,ph]=bode(syss,ww);
mag_ratio=db2mag(mag_db);
for k=1:size(mag_ratio,1)
mag_ratio=mag_ratio(k,:);
end
mag_ratio=mag_ratio.';
for k=1:size(ph,1)
ph=ph(k,:);
end
ph=ph.';
ph=ph-360;
beta=(PM+5)-180;
magbeta=interp1(ph,mag_ratio,beta);
Klag=1/magbeta;
%Lag alpha

lfg=Klag*dcgain(syss);
alphalag=Klow/lfg;
%Lag zero

wwc=interp1(ph,ww,beta);
zlag=-((wwc/10)-(wwc/100));
```

Fragmento de código 3.39. Controlador Lag Network en PMGUIdiscbig

Para ejemplificar el cálculo se utiliza la función de transferencia antes mencionada en la ecuación 3.50, cuya función de transferencia al transformarla al plano continuo se encuentra en la ecuación 3.58, introduciendo como valores de margen de fase mínimo y ganancia de lazo abierto en baja frecuencia mínima los establecidos anteriormente. De esta manera se particularizan las ecuaciones 3.68, 3.69, 3.70, 3.71 y 3.72 en las ecuaciones 3.73, 3.74, 3.75, 3.76 y 3.77 respectivamente.

$$\beta = (110° + 5°) - 180° = -65° \tag{3.73}$$

$$K_{lag} = \frac{1}{3413.6} = 2.9294 \cdot 10^{-4} \tag{3.74}$$

$$plant_{lfg} = (2.9294 \cdot 10^{-4}) \cdot \frac{-11.7 \cdot 0 + 233.9}{0^2 + 3.275 \cdot 0 + 2.339} = 0.0293 \tag{3.75}$$

$$\alpha_{lag} = \frac{10}{0.0293} = 341.362 \tag{3.76}$$

$$z_{lag} = -\left(\frac{0.8973}{10} - \frac{0.8973}{100}\right) = -0.0808 \tag{3.77}$$

Con los resultados de las ecuaciones 3.74, 3.76 y 3.77 puede particularizarse para este caso la ecuación 3.67 en la ecuación 3.78 del controlador Lag calculado en el plano continuo, que se transforma al plano discreto, en la ecuación 3.79, para obtener la función de transferencia del controlador en z. El diagrama PM del sistema controlado se muestra en la figura 3.17, en la que se comprueba que el sistema es ahora estable con un margen de fase de 97.2º, inferior a los 110º establecidos como objetivo pero superior a los 70º, además de reducir el error en régimen estacionario a 0.096, por lo que para este caso la red de atraso diseñada consigue controlar el sistema con robustez.

$$controller = 0.0002929 \frac{s + 0.0808}{s + {0.0808}/{341.362}} = 0.00029294 \frac{s + 0.0808}{s + 0.0002367} \tag{3.78}$$

$$controller = 0.00029412 \frac{z - 0.99196}{z - 0.99998} \tag{3.79}$$

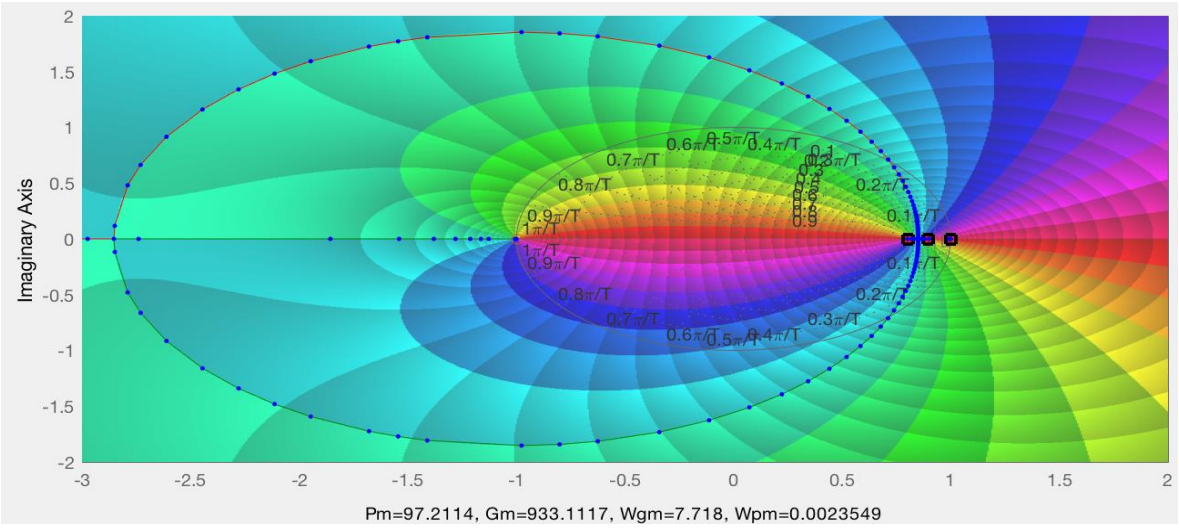

Figura 3.17. Diagrama PM de la función de transferencia en la ecuación 3.50 controlada con controlador Lag Network

Controlador Lead-Lag (Lead-Lag Network)

El controlador o red de adelanto-atraso de fase ejerce una función similar a la del controlador PID en el dominio del tiempo, pero actuando en el dominio de la frecuencia, ya que combinando los efectos de los controladores Lead y Lag reduce el error en régimen estacionario del sistema, además del tiempo de respuesta y la sobreoscilación.

La función de transferencia de un controlador Lead en el plano continuo se encuentra en la ecuación 3.80, por lo que se deberán calcular cinco parámetros: la ganancia de atraso-adelanto del sistema K_{ldlg}, la posición del cero que introduce en el sistema z_{lead} y un coeficiente α_{lead} relacionado con el margen de fase del sistema, ambos correspondientes a la parte Lead del controlador, y la posición del cero que introduce en el sistema z_{lag} y un coeficiente α_{lag} relacionado con el comportamiento en régimen estacionario del sistema, ambos correspondientes a la parte Lag del sistema.

$$controller = K_{ldlg}\frac{\alpha_{lead}(s - z_{lead})}{s - \alpha_{lead}z_{lead}}\frac{(s - z_{lag})/\alpha_{lag}}{s - z_{lag}/\alpha_{lag}} \tag{3.80}$$

Para aplicar el método propuesto en la fuente [21] es necesario fijar unos valores deseados de margen de fase y de error en régimen estacionario, al igual que en el controlador Lead. Se han fijado uno valores de 70º como margen de fase mínimo y 0.01 como error en régimen estacionario máximo debido a que con tales valores se alcanza un sistema razonablemente robusto, siempre siendo posible cambiar estos valores en función de los requisitos del sistema. Además, se transforma la función de transferencia del sistema a controlar al plano s

utilizando la transformada bilineal, resultando la función de transferencia en s representada con la variable local syss.

Tras esta inicialización se calcula la ganancia de adelanto-atraso del sistema, que será calculada de la misma forma que la ganancia K_{lead} del controlador Lead en la ecuación 3.54, calculando previamente la constante de posición del sistema y la ganancia DC de la planta, con las ecuaciones 3.52 y 3.53, calculando finalmente la ganancia con la ecuación 3.81.

$$K_{ldlg} = \frac{K_p}{plant_{lfg}} \tag{3.81}$$

Tras el cálculo de la ganancia se calculan los parámetros de la parte Lead del controlador, que serán calculados de la misma forma en la que ha sido explicada en la parte del trabajo relativa al controlador Lead, con la única diferencia de que la magnitud y la fase ph medidas en la frecuencia central son medidas del sistema multiplicado por la ganancia de adelanto-atraso calculada con la ecuación 3.81, en lugar de ser medidas directamente del sistema sin controlar. Para el cálculo de los factores se utilizan las ecuaciones 3.55, 3.56 y 3.57 antes formuladas, resultando la parte Lead del controlador en la función de transferencia en la ecuación 3.82.

$$G_{clds} = K_{ldlg} \frac{\alpha_{lead}(s - z_{lead})}{s - \alpha_{lead} z_{lead}} \tag{3.82}$$

El cálculo de la parte Lag del controlador se realiza tomando como base la parte Lead ya calculada, midiendo la magnitud y fase del sistema controlado con esa parte en la frecuencia central de 5 rad/s utilizada en el cálculo de la parte Lead. La magnitud medida, almacenada en la variable mag_2, será igual al coeficiente α_{lag}, siguiendo la ecuación 3.83, mientras que la posición del cero z_{lag} será calculada con la ecuación 3.84, en la que ww_1 representa la frecuencia central de 5 rad/s, resultando la parte Lag del controlador en la función de transferencia en la ecuación 3.85. El código de la herramienta en el que se realiza el cálculo de todos los parámetros del controlador se encuentra en el fragmento de código 3.40.

$$\alpha_{lag} = mag_2 \tag{3.83}$$

$$z_{lag} = \frac{-ww_1}{10} \tag{3.84}$$

$$G_{clgs} = \frac{(s - z_{lag})/\alpha_{lag}}{s - z_{lag}/\alpha_{lag}} \tag{3.85}$$

```
%Data

PM=70;
ess=0.01;
Ts=0.1;

syss=d2c(sys,'tustin');

%Lead-Lag gain

Kp_read=1/ess;
plant_lfg=dcgain(syss);
Kldlg=Kp_read/plant_lfg;

%Lead part

ww1=5;
pm_target=PM+7;
[mag1,ph1]=bode(Kldlg*syss,ww1);
del_pm=pm_target-(180+ph1);
alphalead=(1+sin(del_pm*pi/180))/(1-sin(del_pm*pi/180));
zlead=-ww1/sqrt(alphalead);
Gclds=Kldlg*tf(alphalead*[1,-zlead],[1,-alphalead*zlead]);

%Lag part

[mag2,ph2]=bode(Gclds*syss,ww1);
alphalag=mag2;
zlag=-ww1/10;
Gclgs=tf([-1/zlag,1],[-alphalag/zlag,1]);
```

Fragmento de código 3.40. Controlador Lead-Lag Network en PMGUIdiscbig

Para ejemplificar el cálculo se utiliza la función de transferencia antes mencionada en la ecuación 3.50, cuya función de transferencia al transformarla al plano continuo se encuentra en la ecuación 3.58, introduciendo como valores de margen de fase mínimo y error en régimen estacionario máximo los establecidos anteriormente. De esta manera se particularizan las ecuaciones 3.52, 3.53, 3.81, 3.55, 3.56, 3.57, 3.82, 3.83, 3.84 y 3.85 en las ecuaciones 3.86, 3.60, 3.87, 3.88, 3.89, 3.90, 3.91, 3.92, 3.93 y 3.94 respectivamente.

$$K_p = \frac{1}{0.01} - 1 = 99 \tag{3.86}$$

$$K_{ldlg} = \frac{99}{100} = 0.99 \tag{3.87}$$

$$\Delta_{PM} = 77° - (180° + 201.8149°) = -304.8149° \tag{3.88}$$

$$\alpha_{lead} = \frac{1 + sen(-304.8149°)}{1 - sen(-304.8149°)} = 10.1732 \tag{3.89}$$

$$z_{lead} = \frac{-5}{\sqrt{10.1732}} = -1.5676 \tag{3.90}$$

$$G_{clds} = 0.99\frac{10.1732 \cdot (s + 1.5676)}{s + 10.1732 \cdot 1.5676} = 0.99\frac{10.1732 \cdot (s + 1.5676)}{s + 15.9475} \tag{3.91}$$

$$\alpha_{lag} = 27.2328 \tag{3.92}$$

$$z_{lag} = \frac{-5}{10} = -0.5 \tag{3.93}$$

$$G_{clgs} = \frac{(s + 0.5)/27.2328}{s + 0.5/27.2328} = \frac{(s + 0.5)/27.2328}{s + (7.3441 \cdot 10^{-3})} \tag{3.94}$$

Con los resultados de las ecuaciones 3.82 y 3.85 puede particularizarse para este caso la ecuación 3.80 en la ecuación 3.95 del controlador Lead-Lag calculado en el plano continuo, que se transforma al plano discreto, en la ecuación 3.96, para obtener la función de transferencia del controlador en z. El diagrama PM del sistema controlado se muestra en la figura 3.18, en la que se comprueba que el sistema es ahora estable con un margen de fase de 71.27º manteniendo el error en régimen estacionario de 0.0099, por lo que se cumplen ambas especificaciones a la perfección y el ofrece un control correcto y robusto sobre el sistema.

$$controller = 0.99\frac{10.1732 \cdot (s + 1.5676)}{s + 15.9475}\frac{(s + 0.5)}{s + (7.3441 \cdot 10^{-3})}/27.2328 \tag{3.95}$$

$$controller = 0.2272\frac{z - 0.8547}{z - 0.1127}\frac{z - 0.9512}{z - 0.9982} \tag{3.96}$$

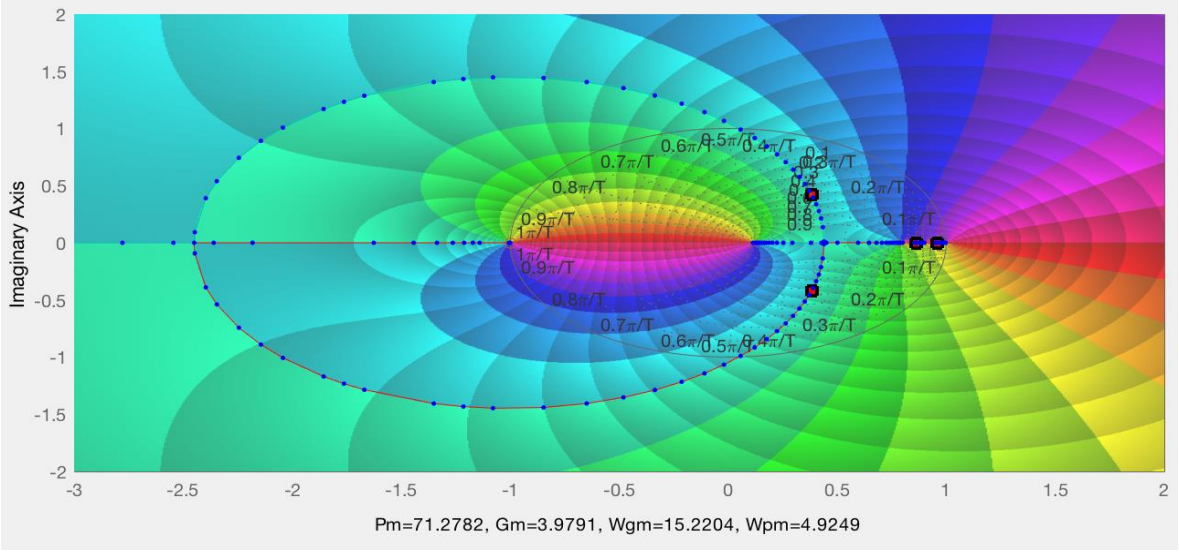

Figura 3.18. Diagrama PM de la función de transferencia en la ecuación 3.50 controlada con controlador Lead-Lag Network

3.5. Inclusión de controladores optimizados con Differential Evolution

El algoritmo de optimización Differential Evolution (DE a partir de ahora), Evolución Diferencial en español, es un método de optimización ubicado en la categoría computación evolutiva creado en 1995 por Rainer Storn y Kenneth Price. Puesto que es un método de computación evolutiva está caracterizado por la codificación de las variables del problema, la búsqueda de una solución óptima relativa a una función de desempeño y la selección de parámetros que influyen en el desempeño del algoritmo. [22]

En concreto el algoritmo DE codifica las D variables del problema en un vector de números reales, utilizando para el cálculo NP padres vectores. Dichos vectores se nombran como x con un subíndice p como índice del individuo con valor entre 1 y NP, y un superíndice g como índice de la generación del individuo con un valor entre 1 y itermax, que identifica el límite superior de iteraciones del cálculo. Cada uno de estos vectores está compuesto por las variables del problema para cada individuo concreto, identificadas como x con superíndice g igual que los padres vectores, pero con subíndice p,m, siendo p el índice del individuo al igual que en los padres vectores y m siendo el índice de variable del individuo, con valor entre 1 y n. Dichas variables tendrán un dominio establecido con los parámetros XVmin y XVmax para la generación aleatoria de la primera generación de vectores.

El algoritmo DE se divide en cuatro pasos, una inicialización que ser realiza al iniciarse el cálculo una sola vez, y tres pasos de mutación, recombinación y selección que se repiten hasta llegar al número de generaciones establecido con el parámetro itermax o hasta que el valor de la función de desempeño a optimizar tenga un valor menor al establecido con el parámetro valor objetivo VTR.

Inicialización

En el primer paso se crea aleatoriamente la primera generación de la población, generando NPxD variables siguiendo la ecuación 3.97, con p tomando valores entre 1 y NP y m tomando valores entre 1 y n, en la que x^{min} y x^{max} con subíndice m representan los valores mínimo y máximo de la variable m, y la función rand(0,1) genera un número aleatorio entre 0 y 1.

$$x_{p,m}^{1} = x_m^{min} + rand(0,1) \cdot (x_m^{max} - x_m^{min}) \tag{3.97}$$

Mutación

Se eligen tres individuos al azar de la población total, denominados vectores objetivos x_a, x_b y x_c, que serán utilizados para generar NP vectores aleatorios ruidosos siguiendo la ecuación 3.98, con p tomando valores entre 1 y NP y con p, a, b y c distintos, en la que F es un parámetro denominado factor de mutación que puede tener valores entre 0 y 2 influyendo en la variabilidad de la solución final.

$$n_p^g = x_c + F \cdot (x_a - x_b) \tag{3.98}$$

Recombinación

Los vectores aleatorios ruidosos con comparados aleatoriamente con los originales variable por variable, generando vectores combinados entre los aleatorios ruidosos y los originales denominados vectores de prueba, siguiendo la ecuación 3.99, con p tomando valores entre 1 y NP y m tomando valores entre 1 y n, en la que CR es un parámetro denominado constante de cruce cuyo valor influirá en la variabilidad de la solución final.

$$t_{p,m}^g = \begin{cases} n_{p,m}^g \; si \; rand(0,1) < CR \\ x_{p,m}^g \; en \; otro \; caso \end{cases} \tag{3.99}$$

Selección

Finalmente se comparan el desempeño de los vectores de prueba y los originales con respecto a la función de desempeño, sustituyendo en la próxima generación los de prueba a los originales en caso de ofrecer mejor desempeño siguiendo la ecuación 3.100, con p tomando valores entre 1 y NP, en la que la función fit() representa la función de desempeño.

$$x_p^{g+1} = \begin{cases} t_p^g \; si \; fit(t_p^g) > fit(x_p^g) \\ x_p^g \; en \; otro \; caso \end{cases} \tag{3.100}$$

Inclusión en la herramienta

En la herramienta se utiliza el método DE para el cálculo de controladores PID reales tomando como función objetivo F a minimizar la suma ponderada entre el número de muestras hasta la estabilización n_s y la sobreoscilación del sistema, en la ecuación 3.101, en la que n_s y M_p representan el número de muestras hasta la estabilización y la sobreoscilación respectivamente.

$$F = 0.3n_s + 0.4M_p \tag{3.101}$$

La función de transferencia de un controlador PID real en el plano discreto se encuentra en la ecuación 3.44, por lo que habrá dos variables cuyo valor debe calcular el algoritmo, la ganancia del controlador K_c y la posición b del cero que introduce el controlador. La función de desempeño se introduce en la herramienta utilizando la función auxiliar tracklsq, cuyo código se encuentra en el fragmento de código 3.41, y que recibe los valores de las variables y devuelve el valor de la función de desempeño para tales valores.

El algoritmo explicado anteriormente en cuatro pasos es realizado con la función devec3, proporcionada por ICSI para aplicar el algoritmo en Matlab, y que para su aplicación requiere la función auxiliar tracklsq en el fragmento de código 3.41 y la asignación de los valores de los parámetros que influirán en la solución final.

Dichos parámetros, en su mayoría explicados anteriormente, son el valor objetivo VTR, el número de variables D, los límites de las variables XV_{min} y XV_{max}, el número de individuos de la población NP, el número máximo de generaciones itermax, el factor de mutación F, la constante de cruce CR, y el parámetro refresh que indica el número de iteraciones tras las cual se muestra por pantalla el resultado en ese momento. Los valores seleccionados para los parámetros D, XV_{min} y XV_{max} son impuestos por el problema, siendo el valor de D 2 al existir dos variables cuyo valor se debe calcular, y [0,0] y [2,2] el de las variables XV_{min} y XV_{max},

al no poder tomar las variables valores negativos. El resto de valores son obtenidos en base a la experiencia y teniendo en cuenta los valores por defecto ofrecidos por el algoritmo.

```
function F = tracklsq(pid,y)
global sys
%% Par?metros del PID
Kc = pid(1);
b = pid(2);

%% Funcion de transferencia del sistema en bucle cerrado
%Gc = tf([Kd, Kp, Ki],[1,0]);
z=tf('z');
Gc=Kc*((z-b)/z)*((z-0.99)/(z-1));
Gp = sys;
Gba = Gc*Gp;
Gbc = feedback(Gba,1);

%% Respuesta del sistema ante entrada escalon unitario
[Y] = step(Gbc);

%% Funcion objetivo
%% Opcion 3: Suma ponderada del factor de sobreoscilacion y el tiempo de
establecimiento
vf = Y(length(Y));
sys2 = stepinfo(Gbc);
ts = sys2.SettlingTime;
Mp = sys2.Overshoot;
F = 0.3*ts+0.4*Mp;
```

Fragmento de código 3.41. Función auxiliar tracklsq

El código con el que se genera el controlador se encuentra en el fragmento de código 3.42, que es complementado con los fragmentos de código 3.27, 3.28 y 3.29 referentes a la inicialización del cálculo, la representación del diagrama PM del sistema controlado y la representación del controlador calculado y de los márgenes de fase y ganancia del sistema controlado respectivamente, variando únicamente el primero de ellos eliminando la parte relativa a los polos deseados seleccionados por el usuario, al no seleccionarse polos para este controlador.

Para ejemplificar el desempeño de esta opción se utilizará como función de transferencia del sistema a controlar la presente en la ecuación 3.17, pese a que el resultado es variable con cada ejecución los valores seleccionados en base a la experiencia para los parámetros consiguen que este converja a un valor prácticamente constante.

Cada vez que se generan el número de generaciones establecido en el parámetro refresh, en este caso 20 generaciones, se muestra por la pantalla de Matlab el resultado en ese momento de la forma observable en la figura 3.19, en la que Iteration indica el número de generación actual y Best, best(1) y best(2) representan los mejor valores actuales para la función de desempeño, la ganancia K_c y la posición del cero b respectivamente.

```
%% Parametros

    VTR = 1.e-6;         % Valor objetivo

    D = 2;               % Par?metros de la poblaci?n

    XVmin = [0 0];     % L?mite inf. de la poblaci?n inicial
    XVmax = [2 2];     % L?mite sup. de la poblaci?n inicial

    y=[0 0];           % Datos del problema

    NP = 40;             % N?mero de candidatos de la poblaci?n

    itermax = 100;        % L?mite superior de iteraciones

    F = 0.8;             % Factor de mutaci?n

    CR = 0.8;            % Costante de cruce

    strategy = 7;         % Tipo de estrategia

    refresh = 20;       % Cada cuantas iteraciones se muestra el resultado

%% Aplicacion del algoritmo Differential Evolution

    [x,f,nf]                                                                    =
devec3('tracklsq',VTR,D,XVmin,XVmax,y,NP,itermax,F,CR,strategy,refresh)
;

% Asignacion de los parametros del PID obtenidos

    Kc = x(1);
    b = x(2);
```

Fragmento de código 3.42. Controlador PID real optimizado con DE en PMGUIdiscbig

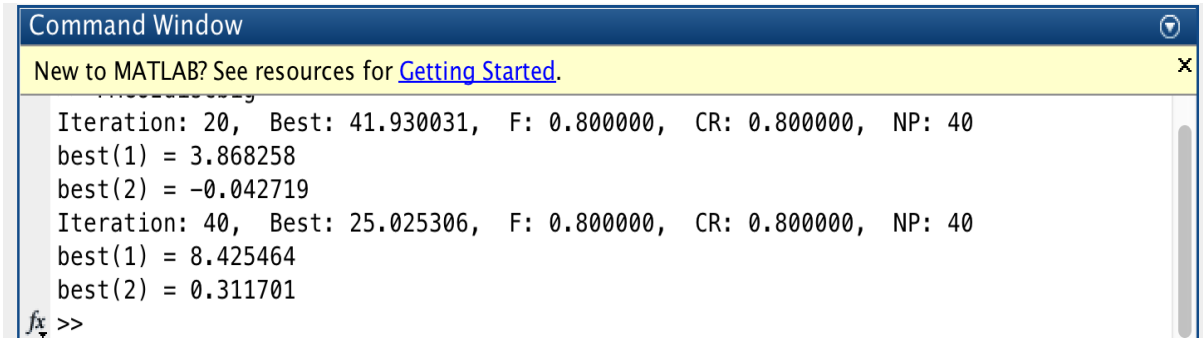

Figura 3.19. Refresco del resultado de la optimización DE en la pantalla de Matlab

La función de transferencia del controlador calculada se encuentra en la ecuación 3.102, mientras que el diagrama PM del sistema controlador puede verse en la figura 3.20, ofreciendo un número de muestras hasta la estabilización de 7 muestras y una sobreoscilación

del 32.31 %, una combinación más estable que las 22 muestras hasta la estabilización y la sobreoscilación del 22.19 % que poseía el sistema antes de ser controlado, como se demuestra al particularizar la ecuación 3.101 para ambos casos en las ecuaciones 3.103 para el sistema antes de ser controlado y 3.104 para el sistema controlado, al mejorar la función de desempeño su valor un 2.97 %.

$$controller = 13.1592\frac{(z-0.3886)}{z}\frac{(z-0.99)}{(z-1)} \tag{3.102}$$

$$F = 0.3 \cdot 22 + 0.4 \cdot 22.19 = 15.48 \tag{3.103}$$

$$F = 0.3 \cdot 7 + 0.4 \cdot 32.31 = 15.02 \tag{3.104}$$

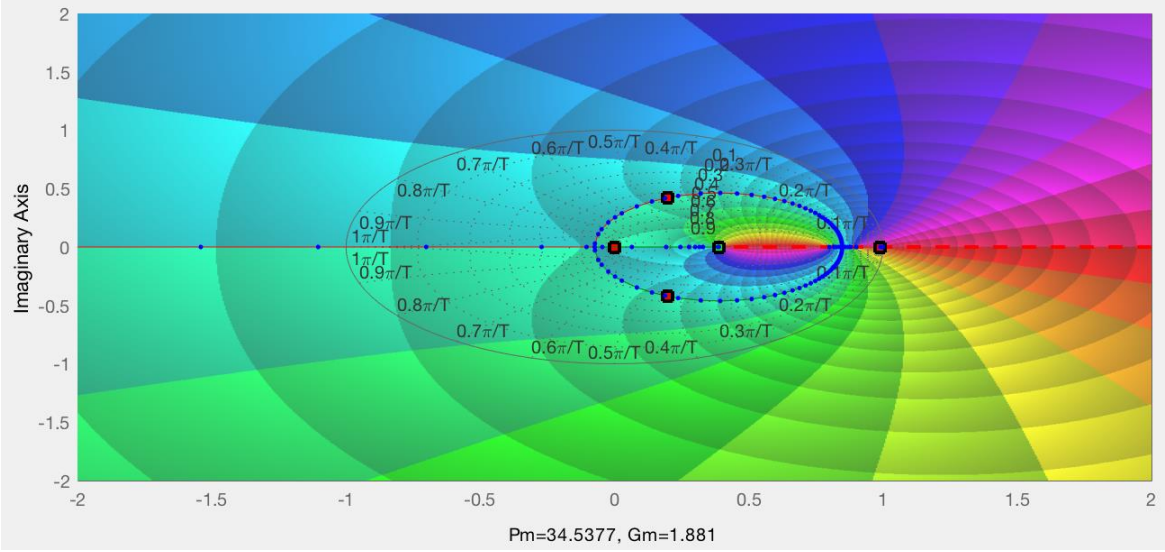

Figura 3.20. Diagrama PM de la función de transferencia en la ecuación 3.17 controlada con controlador PID real optimizado

4. CONCLUSIONES

4.1. Objetivos cumplidos

El cumplimiento de los objetivos del proyecto se mide con respecto a la calidad de los resultados ofrecidos por la herramienta, es decir, en qué medida los sistemas introducidos en la misma para ser controlados lo son realmente con cada tipo de controlador. Por lo tanto, se analizará el rendimiento de la herramienta al intentar controlar diferentes sistemas con diferentes requisitos, ya sean de posición de los polos característicos o de robustez.

Para obtener los resultados se utilizarán cuatro funciones de transferencia distintas como función de transferencia del sistema a controlar, las ya utilizadas en este trabajo en las ecuaciones 3.17 y 3.50, y las representadas en las ecuaciones 4.1 y 4.2. Los datos de los sistemas representados con cada función de transferencia antes de ser controlados se encuentran en la tabla 4.1, en la que p_c representa los polos del sistema en bucle cerrado, n_s el número de muestras de estabilización del sistema, M_p la sobreoscilación del sistema, e_{ss} el error en régimen estacionario del sistema y G_m y P_m los márgenes de ganancia y fase del sistema respectivamente. Los sistemas inestables se identifican puesto que su número de muestras hasta la estabilización y su sobreoscilación son nulas al no estabilizarse nunca, y los márgenes de fase y ganancia son negativos.

$$G(z) = \frac{1}{(z-0.8)(z-0.9)} \tag{4.1}$$

$$G(z) = \frac{(z+0.4)}{(z+0.2)(z-0.6)} \tag{4.2}$$

Función de transferencia del sistema a controlar	**p_c**	**n_s (muestras)**	**M_p (%)**	**e_{ss} (%)**	**G_m (dB)**	**P_m (º)**
Ecuación 3.17	0.8 ± 0.28i	22	22.19	16.67	30.68	55.64
Ecuación 3.50	0.35 ± 1.26i	-	-	0.99	-11.06	-30.63
Ecuación 4.1	0.85 ± 1i	-	-	1.96	-11.06	-50.13
Ecuación 4.2	-0.3 ± 0.44i	5	34.29	25.53	6.58	56.48

Tabla 4.1. Datos de los sistemas a controlar para probar la herramienta

Controladores básicos

Los primeros controladores cuyo rendimiento se analiza son los controladores básicos P, PI, PD real y PID real, para lo que se requiere una selección previa de la posición de los polos deseados por el usuario. Se seleccionan cuatro posiciones variando las coordenadas de los polos en los ejes real e imaginario, obteniendo como posición de los polos deseados 0.5 ± 0.5i, 1 ± 0.5i, -0.5 ± 0.5i y 0.5 ± 1i. Además, puesto que el controlador proporcional solo ofrece buenos resultados cuando la posición de los polos deseados se encuentra incluida en el lugar de las raíces del sistema antes de ser controlado, se selecciona una posición perteneciente al lugar de las raíces para cada función de transferencia, siendo estas posiciones 0.22 ± 0.82i, -0.08 ± 1.6i, 0.85 ± 0.5i y -0.16 ± 0.38i para las funciones de transferencia en las ecuaciones 3.17, 3.50, 4.1 y 4.2 respectivamente.

Los resultados para el controlador P se encuentran en la tabla 4.2, obtenidos introduciendo como polos deseados por el usuario para cada función de transferencia de la tabla 4.1 los mencionados en el anterior párrafo para el controlador proporcional. Se puede comprobar que en los cuatro casos el control es correcto ya que la posición de los polos característicos del sistema una vez controlado, p_c, es la misma que la introducida como posición de los polos deseados p_d. Además, en la tabla se incluye la función de transferencia del controlador calculado.

Función de transferencia del sistema a controlar	**p_d**	**Función de transferencia del controlador**	**p_c**
Ecuación 3.17	0.22 ± 0.82i	12.68	0.22 ± 0.82i
Ecuación 3.50	-0.08 ± 1.6i	1.85	-0.08 ± 1.6i
Ecuación 4.1	0.85 ± 0.5i	0.25	0.85 ± 0.5i
Ecuación 4.2	-0.16 ± 0.38i	0.72	-0.16 ± 0.38i

Tabla 4.2. Resultados del controlador P

Para probar los controladores PI se introducen las mismas cuatro ecuaciones como funciones de transferencia del sistema a controlar, sin variar la posición de los polos deseados al no influir esta en el control ejercido. En la tabla 4.3 se encuentran los resultados para el controlador PI, incluyendo la función de transferencia del controlador calculado y los valores del número de muestras hasta la estabilización n_s y de la sobreoscilación M_p del sistema controlado, incluyendo entre paréntesis la diferencia con respecto a los valores de los sistemas antes de ser controlados en porcentaje, con valor - en caso de que el sistema controlado sea inestable en bucle cerrado. No es necesario incluir el error en régimen estacionario ya que al introducir el controlador integral un polo en 1 anula dicho error. En los casos en los que el sistema es controlado correctamente, es decir aquellos en la que el

comportamiento dinámico del sistema no varía y se controla el error en régimen permanente, se señalan en verde en la tabla, señalándose en rojo en el caso contrario.

Función de transferencia del sistema a controlar	**Función de transferencia del controlador**	**n_s (muestras)**	**M_p (%)**
Ecuación 3.17	(z-0.97)/(z-1)	59 (+168.18)	12.73 (-42.63)
Ecuación 3.50	(z-0.89)/(z-1)	-	-
Ecuación 4.1	(z-0.98)/(z-1)	-	-
Ecuación 4.2	(z-0.88)/(z-1)	25 (+400)	0 (-100)

Tabla 4.3. Resultados del controlador PI

En la tabla 4.3 se comprueba que el controlador PI ofrece buenos resultados para todos los casos manteniendo las propiedades dinámicas del sistema, reduciendo además siempre el error en régimen estacionario del sistema a cero al introducir un polo en 1.

Para probar los controladores PD real y PID real se utilizan las cuatro posiciones antes mencionadas como posición de los polos deseados para cada una de las funciones de transferencia de la tabla 4.1. En las tablas 4.4 y 4.5 se encuentran los resultados para los controladores PD real y PID real respectivamente, incluyendo la posición seleccionada de polos deseados p_d, la función de transferencia del controlador calculado, la posición de los polos en bucle cerrado del sistema controlado p_c, los valores del número de muestras n_s y de la sobreoscilación M_p del sistema controlado, incluyendo entre paréntesis la diferencia con respecto a los valores de los sistemas antes de ser controlados en porcentaje, con valor - en caso de que el sistema controlado sea inestable en bucle cerrado, y el error en régimen estacionario del sistema controlado e_{ss}., con su diferencia respecto al valor en el sistema sin controlar en porcentaje entre paréntesis en el caso del controlador PD real, ya que al introducir los otros dos un polo en 1 hacen el error en régimen estacionario del sistema nulo. En los casos en los que el sistema es controlado correctamente, es decir aquellos en la que la posición de los polos característicos del sistema controlado corresponde con la posición seleccionada de polos deseados, se señalan en verde en la tabla, señalándose en rojo en el caso contrario.

Función de transferencia del sistema a controlar	p_d	Función de transferencia del controlador	p_c	n_s (muestras)	M_p (%)	e_{ss} (%)
Ecuación 3.17	0.5 ± 0.5i	7*(z-0.31)/z	0, 0.5 ± 0.5i	12 (-45.45)	36.46 (+64.31)	4 (-76)
	1 ± 0.5i	3*(z-1.77)/z	0, 0.15, 1.25	-	-	9.52 (-42.89)
	-0.5 ± 0.5i	27*(z-0.08)/z	0, -0.5 ± 0.5i	11 (-50)	172.18 (+675.94)	0.8 (-95.2)
	0.5 ± 1i	7*(z+0.78)/z	0, 0.5 ± 1i	-	-	1.6 (-90.4)
Ecuación 3.50	0.5 ± 0.5i	0.33*(z-0.55)/z	0.37, 0.5 ± 0.5i	13	39.34	6.33 (-62.03)
	1 ± 0.5i	0.1*(z-2.33)/z	1.29, 0.15 ± 0.41i	-	-	7.73 (-53.63)
	-0.5 ± 0.5i	3.14*(z+0.93)/z	-0.46, -0.49 ± 2.47i	-	-	0.16 (-99.04)
	0.5 ± 1i	0.65*(z-0.1)/z	0.05, 0.5 ± 1i	-	-	1.69 (-89.86)
Ecuación 4.1	0.5 ± 0.5i	0.48*(z-0.73)/z	0.7, 0.5 ± 0.5i	13	34.1	13.33 (-20.04)
	1 ± 0.5i	0.07*(z-5.36)/z	1.32, 0.19 ± 0.5i	-	-	7.02 (-57.89)
	-0.5 ± 0.5i	2.92*(z+0.46)/z	-0.32, 1.01 ± 1.81i	-	-	0.47 (-97.18)
	0.5 ± 1i	1.23*(z-0.71)/z	0.7, 0.5 ± 1i	-	-	5.33 (-68.03)
Ecuación 4.2	0.5 ± 0.5i	0.28*(z-1.47)/z	0.89, -0.36 ± 0.25i	24 (+380)	45 (+31.23)	161.23 (+867.19)
	1 ± 0.5i	1.24*(z-0.92)/z	-1.16, -0.49, 0.81	-	-	77.55 (+365.21)
	-0.5 ± 0.5i	1.49*(z+0.08)/z	-0.09, -0.5 ± 0.5i	10 (+100)	81.06 (+136.4)	17.58 (+5.46)
	0.5 ± 1i	0.24*(z-4.55)/z	1.3, -0.57 ± 0.14i	-	-	64.89 (+289.26)

Tabla 4.4. Resultados del controlador PD real

En la tabla 4.4 se comprueba que el controlador PD real ofrece buenos resultados para exactamente la mitad de los casos probados, reduciendo generalmente el número de muestras hasta la estabilización y el error en régimen estacionario del sistema a costa de aumentar la sobreoscilación. Los casos en los que el control no es correcto, al igual que en los resultados del controlador PI, se debe a la posición relativa de los polos deseados respecto a la de los polos característicos del sistema antes de ser controlado.

Función de transferencia del sistema a controlar	p_d	Función de transferencia del controlador	p_c	n_s (muestras)	M_p (%)
Ecuación 3.17	0.5 ± 0.5i	7.07*((z-0.31)/z)*((z-0.99)/(z-1))	0, 0.99, 0.5 ± 0.51i	57 (+159.09)	32.93 (+64.31)
	1 ± 0.5i	3*((z-1.77)/z)*((z-0.99)/(z-1))	0, 0.16, 0.99, 1.25	-	-
	-0.5 ± 0.5i	27.16*((z-0.08)/z)*((z-0.99)/(z-1))	0, 0.99, -0.5 ± 0.51i	14 (-36.36)	171.6 (+675.94)
	0.5 ± 1i	7.03*((z+0.78)/z)*((z-0.99)/(z-1))	0, 0.99, 0.5 ± 1.01i	-	-
Ecuación 3.50	0.5 ± 0.5i	0.34*((z-0.55)/z)*((z-0.99)/(z-1))	0.36, 0.99, 0.5 ± 0.51i	106	33.46
	1 ± 0.5i	0.1*((z-2.33)/z)*((z-0.99)/(z-1))	0.99, 1.29, 0.16 ± 0.39i	-	-
	-0.5 ± 0.5i	3.16*((z+0.93)/z)*((z-0.99)/(z-1))	-0.46, 0.99, -0.49 ± 2.48i	-	-
	0.5 ± 1i	0.65*((z-0.1)/z)*((z-0.99)/(z-1))	0.05, 0.99, 0.5 ± 1i	-	-
Ecuación 4.1	0.5 ± 0.5i	0.48*((z-0.73)/z)*((z-0.99)/(z-1))	0.7, 0.99, 0.5 ± 0.5i	206	17.54
	1 ± 0.5i	0.07*((z-5.36)/z)*((z-0.99)/(z-1))	0.99, 1.32, 0.19 ± 0.5i	-	-
	-0.5 ± 0.5i	2.94*((z+0.46)/z)*((z-0.99)/(z-1))	-0.31, 0.99, 1.01 ± 1.81i	-	-
	0.5 ± 1i	1.23*((z-0.71)/z)*((z-0.99)/(z-1))	0.7, 0.99, 0.51 ± 1i	-	-
Ecuación 4.2	0.5 ± 0.5i	0.28*((z-1.47)/z)*((z-0.99)/(z-1))	0.84, 1.01, -0.36 ± 0.25i	-	-
	1 ± 0.5i	1.24*((z-0.92)/z)*((z-0.99)/(z-1))	-1.15, -0.49, 0.8, 1	-	-
	-0.5 ± 0.5i	1.5*((z+0.08)/z)*((z-0.99)/(z-1))	-0.1, 0.99, -0.5 ± 0.5i	259 (+5080)	50 (+45.82)
	0.5 ± 1i	0.25*((z-4.55)/z)*((z-0.99)/(z-1))	0.98, 1.32, -0.58 ± 0.13i	-	-

Tabla 4.5. Resultados del controlador PID real

En la tabla 4.5 se comprueba que el controlador PID real ofrece resultados muy similares al controlador PD real, debido a que el método para el cálculo de ambos es muy similar, con la diferencia de básica de la introducción del cero en 0.99 y el polo en 1, además de una leve variación de la ganancia. Este polo en 1 anula el error en régimen estacionario del sistema, pero produce un desempeño más lento del mismo, como puede comprobarse en la tabla ya que tanto el número de muestras hasta la estabilización como la sobreoscilación del sistema suelen incrementarse con respecto a los sistemas antes de ser controlados, sin llegar a ser inestables algunos de ellos debido a la acción compensatoria de ese polo ejercida por el cero en 0.99.

Controladores Lead, Lag y Lead-Lag

Las redes de adelanto y atraso se utilizan generalmente para el control de sistemas de forma robusta, lo cual es medible comprobando el margen de fase del sistema y el error en régimen estacionario, si el margen de fase es mayor que 70º y el error en régimen estacionario es menor del 10 % podemos considerar el sistema como robusto y por lo tanto el objetivo buscado con la inclusión de este tipo de controladores será satisfecho correctamente.

Para obtener resultados se utilizan como función de transferencia del sistema a controlar las mismas cuatro ecuaciones 3.17, 3.50, 4.1 y 4.2 utilizadas para obtener resultados para los controladores básicos, sin introducción de polos deseados ya que no son utilizados en el cálculo. Se seleccionan los mismos parámetros para los controladores que los elegidos en el punto 3.4 al explicar este tipo de controladores, es decir, para el controlador Lead se selecciona un margen de fase mínimo de 70º y un error en régimen estacionario máximo del 10 %, para el controlador Lag se selecciona un margen de fase mínimo de 110º y una ganancia de lazo abierto en baja frecuencia máxima de 10, lo que supone un error en régimen estacionario máximo el 9 %, y para el controlador Lead-Lag se selecciona un margen de fase mínimo de 70º y un error en régimen estacionario máximo del 1 %, ya que este último tipo de controlador permite reducir más dicho error al combinar la red de adelanto y de atraso como fue explicado en el punto 3.4.

En las tablas 4.6, 4.7 y 4.8 se incluyen los resultados de los controladores Lead, Lag y Lead-Lag respectivamente para cada función de transferencia del sistema a controlar, incluyendo la función de transferencia del controlador obtenido, además de los márgenes de ganancia y fase del sistema controlado, G_m y P_m, y el error en régimen estacionario del sistema controlado, e_{ss}, incluyendo además entre paréntesis la variación en porcentaje con respecto a los valores del sistema antes de ser controlado.

Al igual que en las tablas anteriores de los controladores básicos, en los casos en los que el sistema es controlado correctamente, es decir aquellos en los que el margen de fase es superior a 70º y en el error en régimen estacionario es inferior al 10 %, se señalan en verde en la tabla, señalándose en rojo en el caso contrario.

Función de transferencia del sistema a controlar	**Función de transferencia del controlador**	**P_m (º)**	**G_m (dB)**	**e_{ss} (%)**
Ecuación 3.17	6.27*(z-0.8)/(z-0.29)	58.7 (+5.5)	11.86 (-61.34)	10 (-40.01)
Ecuación 3.50	0.55*(z-0.85)/(z-0.11)	28.5 (+193.05)	4.35 (+139.33)	10 (+910.1)
Ecuación 4.1	2.37*(z-0.91)/(z+0.15)	Inf	-6.51 (+41.14)	Inf
Ecuación 4.2	0.90*(z-0.24)/(z-1)	40.39 (-28.49)	10.21 (+55.17)	8.95 (-64.94)

Tabla 4.6. Resultados del controlador Lead Network

En la tabla 4.6 puede comprobarse que el control de ambas variables no consigue realizarse correctamente utilizando únicamente el control Lead, ya que se prioriza el ajuste del error en régimen estacionario del sistema en 10 %, no consiguiendo incrementar la fase del sistema hasta los 70º preestablecidos.

Función de transferencia del sistema a controlar	**Función de transferencia del controlador**	**P_m (º)**	**G_m (dB)**	**e_{ss} (%)**
Ecuación 3.17	0.68*(z-0.99)/(z-1)	67.66 (+21.6)	34.11 (+11.18)	9.17 (-44.99)
Ecuación 3.50	2.9e-4*(z-0.99)/(z-1)	97.21 (+417.37)	59.39 (+636.98)	9.64 (+873.74)
Ecuación 4.1	0.02*(z-0.99)/(z-1)	135.5 (+370.3)	25.07 (+326.67)	9.11 (+364.8)
Ecuación 4.2	0.8*(z-0.95)/(z-0.99)	73.2 (+29.6)	8.63 (+31.16)	9.1 (-64.36)

Tabla 4.7. Resultados del controlador Lag Network

En la tabla 4.7 se observa que en todos los casos menos en el primero de ellos el sistema es controlado a la perfección llegando en algunos casos incluso a aumentar en torno a un 400 % el margen de fase del sistema. En el primer caso el margen de fase conseguido se encuentra cercano al valor deseado de 70º, por lo que puede asumirse que incrementando el parámetro de margen de fase mínimo del sistema de 110º a 120º se alcanzaría el valor deseado de margen de fase para todos los sistemas a controlar.

Función de transferencia del sistema a controlar	**Función de transferencia del controlador**	**P_m (º)**	**G_m (dB)**	**e_{ss} (%)**
Ecuación 3.17	3.64*((z-0.8)/(z-0.29)((z-0.95)/(z-1))	71.4 (+28.32)	16.78 (-45.31)	1 (-94)
Ecuación 3.50	0.23*((z-0.85)/(z-0.11)((z-0.95)/(z-1))	71.3 (+332.78)	12 (+208.5)	1 (+1.01)
Ecuación 4.1	0.56*((z-0.91)/(z+0.15)((z-0.95)/(z-1))	71 (+241.63)	6.18 (+155.88)	1.01 (-48.47)
Ecuación 4.2	0.31*((z-0.3)/(z-0.79)((z-0.95)/(z-1))	71.3 (+26.24)	19.56 (+197.26)	1 (-96.08)

Tabla 4.8. Resultados del controlador Lead-Lag Network

En la tabla 4.8 puede observarse que en todos los casos el controlador Lead-Lag consigue ajustar la respuesta del sistema a los parámetros introducidos en el controlador con precisión, manteniendo con independencia del sistema a controlar el margen de fase en torno a 70º y el error en régimen estacionario en torno a 1. Debido al excelente desempeño del controlador al combinar una red de adelanto y una de atraso, con respecto al control ejercido por estas por separado, hace que se recomiende su uso en el caso de requerir un control robusto.

Controlador PID real optimizado con DE

Puesto que la optimización del controlador se realiza en cuanto al número de muestras de estabilización y la sobreoscilación del sistema como muestra la ecuación 3.106, la validez o no del control realizado se medirá con respecto a estos parámetros.

En la tabla 4.9 se muestran los resultados del controlador optimizado tomando como funciones del sistema controlar las ecuaciones 3.17, 3.50, 4.1 y 4.2, incluyendo la función de transferencia obtenida para el controlador, el número de muestras hasta la estabilización n_s, la sobreoscilación M_p, y la suma ponderada de ambas utilizando la ecuación 3.106 del sistema controlado, incluyendo entre paréntesis sus variación en porcentaje con respecto a los valores del sistema antes de ser controlado.

A diferencia del resto de tablas, al no introducir unos parámetros límites para evaluar la respuesta del sistema, no se puede diferenciar claramente entre sistemas correctamente controlados y sistemas no controlados, por lo que en lugar de diferenciar con colores entre ambas respuestas se analizará su comportamiento con respecto a los sistemas antes de ser controlados.

Función de transferencia del sistema a controlar	**Función de transferencia del controlador**	**n_s (muestras)**	**M_p (%)**	**Suma ponderada de n_s y M_p**
Ecuación 3.17	13.16*((z-0.39)/z)*((z-0.99)/(z-1))	7 (-68.18)	32.31 (+45.61)	15.02 (-2.97)
Ecuación 3.50	0.58*((z-0.55)/z)*((z-0.99)/(z-1))	50 (-)	64.26 (-)	40.7 (-)
Ecuación 4.1	0.57*((z-0.52)/z)*((z-0.99)/(z-1))	117 (-)	68.22 (-)	62.39 (-)
Ecuación 4.2	1.39*((z+0.47)/z)*((z-0.99)/(z-1))	227 (+4440)	38.74 (+12.98)	83.6 (+449.28)

Tabla 4.9. Resultados del controlador PID real optimizado

Observando la tabla 4.9 puede comprobarse que para el primer caso, como ya fue explicado el punto 3.5, el control mejora el sistema mínimamente debido a que el sistema antes de ser controlado ya mostraba buenas características en cuanto a número de muestras hasta la estabilización y sobreoscilación, mientras que en el caso de la ecuación 4.2 la respuesta del sistema controlado es ralentizada con respecto a la del sistema antes de ser controlado, lo que es debido a que los polos y ceros de este sistema están muy alejados de la posición 1, lo que hace que al introducir un polo y un cero cerca de dicha posición el sistema se desestabilice provocando sobre todo el alto valor del número de muestras hasta la estabilización. Por último, de los otros dos casos no se tenían valores de respuesta antes de ser controlados debido a que eran inestables, lo cual es solucionado con el control, no siendo posible comparar la respuesta del sistema controlado con la previa a ser controlado.

4.2. Impacto socio-económico

Para realizar un análisis económico del proyecto se valora la posibilidad de comercializar la herramienta desarrollada como una herramienta adicional de pago para Matlab.

El primer punto a tener en cuenta es el hecho de que la licencia de la que dispone la Universidad Carlos III de Madrid la cual ha sido utilizada para realizar este proyecto no permite la comercialización de las aplicaciones desarrolladas utilizándola, al ser una licencia de tipo académica, por lo que sería necesario adquirir una licencia estándar por valor de 2000 €, además de la adquisición de un compilador que permita comercializar el trabajo por valor de 5000 €, lo que hace un total de coste relativo a la licencia de 7000 €. [24]

La segunda parte a estudiar para realizar el análisis económico es la valoración de la fuerza de trabajo invertida en el proyecto. El proyecto ha sido realizado por un único trabajador con una inversión de tiempo estimada de 350 horas, por lo que valorando personalmente en 15 € el coste de cada una de esas horas supone un valor del trabajo realizado de 5250 €.

Además se debe tener en cuenta la amortización del ordenador portátil utilizado para el desarrollo del proyecto y el uso de la licencia Matlab académica proporcionada por la universidad durante las 350 horas de trabajo, cuyo equivalente en días es 14.58 días. En la actualidad, el porcentaje máximo anual de amortización para equipos electrónicos es del 20 % según la página web de la Agencia Tributaria del Gobierno de España [25], lo que aplicado a los 14.58 días de trabajo y teniendo en cuenta que el coste del ordenador portátil utilizado es de 1506 €, supone una amortización de 12.03 €, como es calculado en la ecuación 4.3.

Por otro lado la licencia Matlab de la que dispone la universidad tiene un coste de 500 € anuales por persona [24], lo que aplicado a los 14.58 días de trabajo supone un coste de licencia de 19.97 €, como se muestra en la ecuación 4.4.

$$Coste\ de\ amortización = 0.2 \cdot 1506 \cdot \frac{14.58}{365} = 12.03\ € \tag{4.3}$$

$$Coste\ de\ licencia\ actual = 500 \cdot \frac{14.58}{365} = 19.97\ € \tag{4.4}$$

En la tabla 4.10 se muestra el presupuesto completo del proyecto, con un valor de 12282€.

Presupuesto completo del proyecto	
Coste de licencias a adquirir	7000 €
Coste del trabajo realizado	5250 €
Coste de amortización	12.03 €
Coste de licencia actual	19.97 €
Coste total	**12282 €**

Tabla 4.10. Presupuesto completo del proyecto

Esto hace que el valor a generar para que la herramienta sea rentable es de un mínimo de 12282 €. El precio ofrecido por Matlab para todos sus conjuntos de herramientas propios de pago es de 1000 €, por lo que el coste de una única herramienta puede oscilar entre los 100 y los 250 €. Estableceremos este último valor como precio de venta de la herramienta.

En la gráfica de la figura 4.1 se muestra la relación entre el número de clientes que ha adquirido la herramienta y las ganancias obtenidas con una línea azul oscura representando en el eje x el número de clientes y en el eje y la ganancia económica obtenida en € , señalando además el valor de ganancias a partir del cual la herramienta ha generado beneficio con una línea naranja, 12282 €.

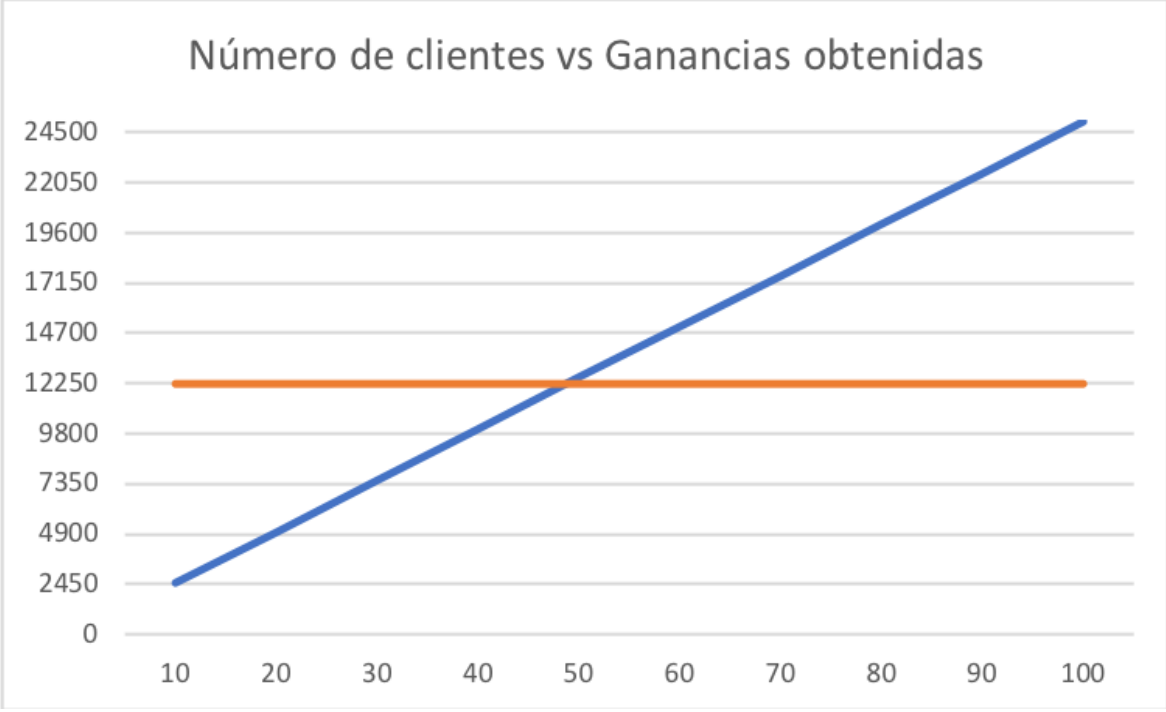

Figura 4.1. Análisis económico del proyecto

Observando la gráfica en la figura 4.1 puede comprobarse que a partir de 50 clientes la herramienta es rentable, un número de clientes difícil de alcanzar al ofrecer Matlab numerosas herramientas propias relacionadas con la ingeniería de control a un coste mucho

menor que el propuesto en este análisis, por lo que la opción de la comercialización de la herramienta es descartada, lo cual es lógico debido a que el propósito de este proyecto es con fines académicos y de investigación.

4.3. Líneas futuras de trabajo

Debido a que este es el primer trabajo del Departamento de Ingeniería de Sistemas y Automática de la Universidad Carlos III de Madrid en el que se aplican los diagramas PM a sistemas discretos, aún no han sido exploradas todas las posibilidades que ofrece, ya que en este trabajo han sido utilizados como medio de análisis para la evaluación del control realizado por los diferentes controladores, existiendo la posibilidad de utilizar este tipo de diagramas directamente para el diseño de controladores enfocados al control robusto controlando el margen de fase del sistema.

Un ejemplo de este nuevo enfoque podría ser la inclusión de un nuevo controlador optimizado con el algoritmo DE, pero en lugar de optimizar la respuesta del sistema en el rango del tiempo, hacerlo en el rango de la frecuencia utilizando los diagramas PM. Utilizando el código de Matlab podría medirse en el propio diagrama PM el margen de fase del sistema y optimizar cual debería ser la posición de los polos en bucle cerrado del sistema para obtener un margen de fase concreto consiguiendo además fase plana en la frecuencia en la que se alcanza este margen de fase.

Otra posible línea futura de trabajo puede ser la inclusión de controladores de orden fraccionario, los llamados controladores FOPID, en la herramienta, ya que dichos controladores tienen un rango de validez mayor que los controladores básicos incluidos en este proyecto, permitiendo controlar un mayor número de posibles sistemas con la misma herramienta.

BIBLIOGRAFÍA

[1] R. Piedrafita Moreno, "Evolución Histórica de la Ingeniería de Control", Tesis doctoral, Dpto. de Ingeniería en Sistemas y Automática, Escuela Universitaria de Ingeniería Técnica Industrial de Zaragoza, Zaragoza, España, 1999. [En línea]. Disponible en http://automata.cps.unizar.es/regulacionautomatica/historia.PDF

[2] E. Lee, "Self-regulating wind machine", Gran Bretaña Patente 615, 1745.

[3] T. Mead, "Regulator for wind and other mills", Gran Bretaña Patente 1628, 1787.

[4] G. B. Airy, "On the regulator of the clock-work for effecting uniform movement of equatoreals", *Monthly Notices of the Royal Astronomical Society*, vol. 11, pp. 249.267, enero 1840.

[5] J. C. Maxwell, "On Governors", *Proceedings of the Royal Society of London*, vol. 16, pp. 270-283, enero 1868.

[6] E. J. Routh, *A treatise on the stability of a given state of motion*. Londres: Macmillan, 1877.

[7] A. Hurwitz, "Uber die Bedingungen, un ter welchen eine Gleichung nur Wurzelm mit negative reelen Teilen besitzt", *Mathematische Annalen*, vol. 46, pp. 273-284, enero 1885.

[8] O. Heaviside, *Electromagnetic Theory*. Londres: The Electrician, 1899.

[9] H. Nyquist, "Regeneration theory", *Bell System Technical Journal*, vol. 11, pp. 126-147, enero 1932.

[10] H. S. Black, "Stabilized feedback amplifiers", *Bell System Technical Journal*, vol. 13, pp. 1-18, enero 1934.

[11] H. W. Bode, "Relations between attenuation and phase in feedback amplifier design", *Bell System Technical Journal*, vol. 19, pp. 421-454, enero 1940.

[12] W. R. Evans, "Graphical Analysis of Control Systems", *Transactions of the AIEE*, vol. 67, pp. 547-551, enero 1948.

[13] W. R. Evans, "Control System Synthesis by Root Locus Method", *Transactions of the AIEE*, vol. 69, pp. 1-4, enero 1950.

[14] J. G. Truxal, *Feedback theory and control system synthesis*. Nueva York: McGraw Hill, 1954.

[15] E. I. Jury, *Sampled-Data Control Systems*. Berkeley: University of California, 1958.

[16] J. G. Ziegler y N. B. Nichols, "Optimum settings for automatic controllers", *ASME*, vol. 64, pp. 759-768, enero 1942.

[17] R. E. Kalman, “Contributions to the theory of optimal control”, *Boletín de la Sociedad Matemática Mexicana*, vol. 5, pp. 102-119, enero 1960.

[18] M. Améstegui Moreno, “Apuntes de Control PID”, Dpto. de Sistemas de Control, Universidad Mayor de San Andrés, La Paz, Bolivia, enero 2001. [En línea]. Disponible en https://www.info-transistor.info/biblioteca/Control%20Pid.pdf

[19] Y. Li, K. H. Ang y G. C. Y. Chong, “PID control system analysis and design”, *IEEE Control Systems Magazine*, vol. 26, pp. 32-41, noviembre 2007.

[20] S. Garrido y L. Moreno, “PM Diagram of the Transfer Function and Its Use in the Design of Controllers”, *Journal of Mathematics and System Science*, vol. 5, pp. 138-149, enero 2015.

[21] J. H. Chow, D. K. Frederick y N. W. Chbat, *Discrete-Time Control Problems Using Matlab*, 1ª ed. California: Brookware Companion Series, 2002.

[22] R. Storn y K. Price, "Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces", ICSI, U.S., Informe Técnico TR-95-012, 1995.

[23] J. L. Gil Ibáñez, “El contrato de licencia de nuevas tecnologías”, *Iberley*, 16-10-2017. [En línea]. Disponible en https://www.iberley.es/temas/contrato-licencia-nuevas-tecnologias-44751

[24] SOFTIN, “Descripción Matlab”, *softin.info*, 24-5-2015. [En línea]. Disponible en http://softin.info/matlab-long/

[25] Agencia Tributaria del Gobierno de España, “Tabla de coeficientes de amortización lineal”, *Agencia Tributaria*, 1-1-2015. [En línea]. Disponible en http://www.agenciatributaria.es/AEAT.internet/Inicio/_Segmentos_/Empresesas_y_profesionales/Empresas/Impuesto_sobre_Sociedades/Periodos_impositivos_a_partir_de_1_1_2015/Base_imponible/Amortizacion/Tabla_de_coeficientes_de_amortizacion_lineal_.shtml

ANEXO. DIAGRAMAS DE FLUJO

Diagrama de flujo del cálculo de controladores básicos (Punto 3.3)

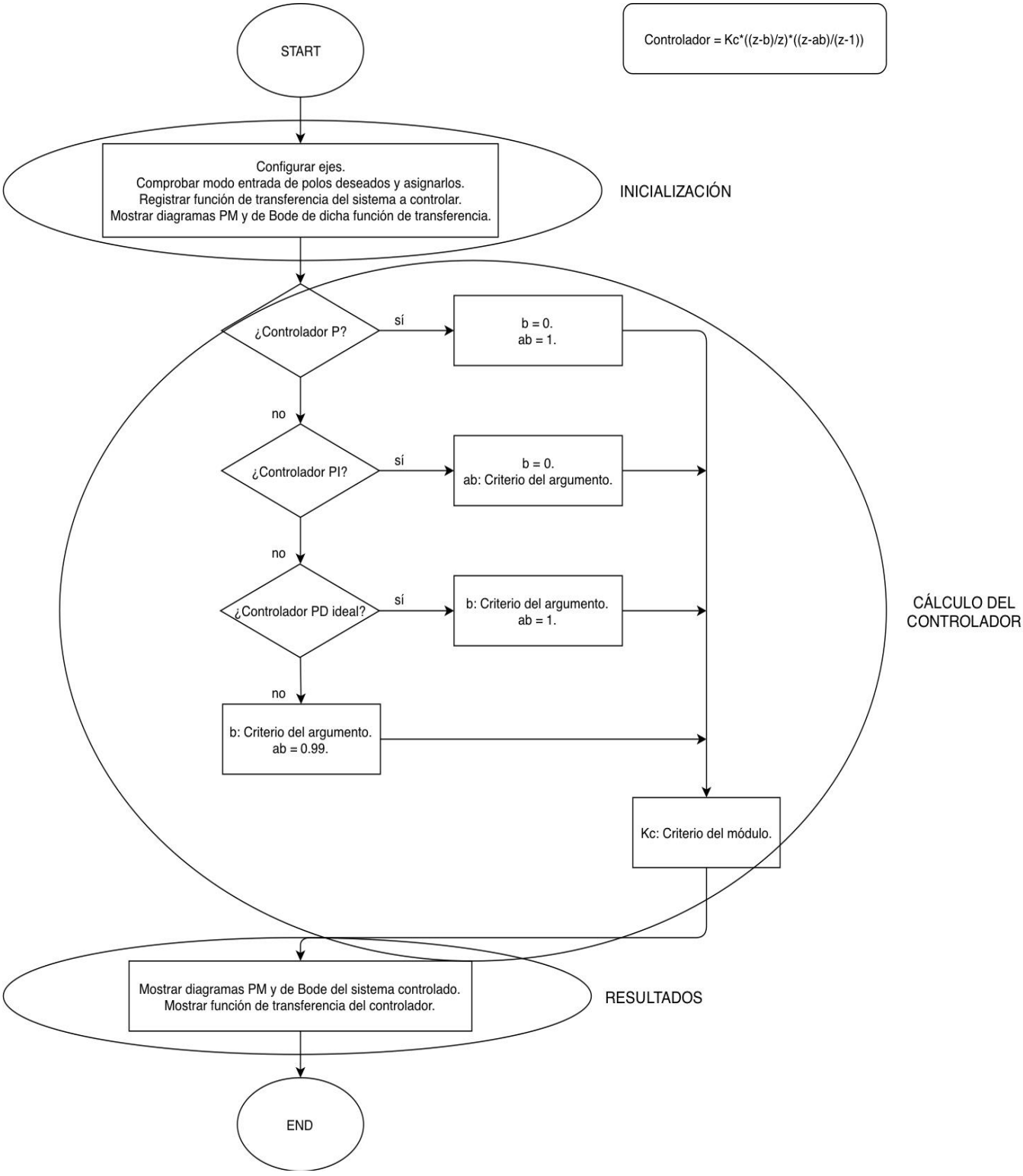

Diagrama de flujo del cálculo de redes de atraso y adelanto (Punto 3.4)

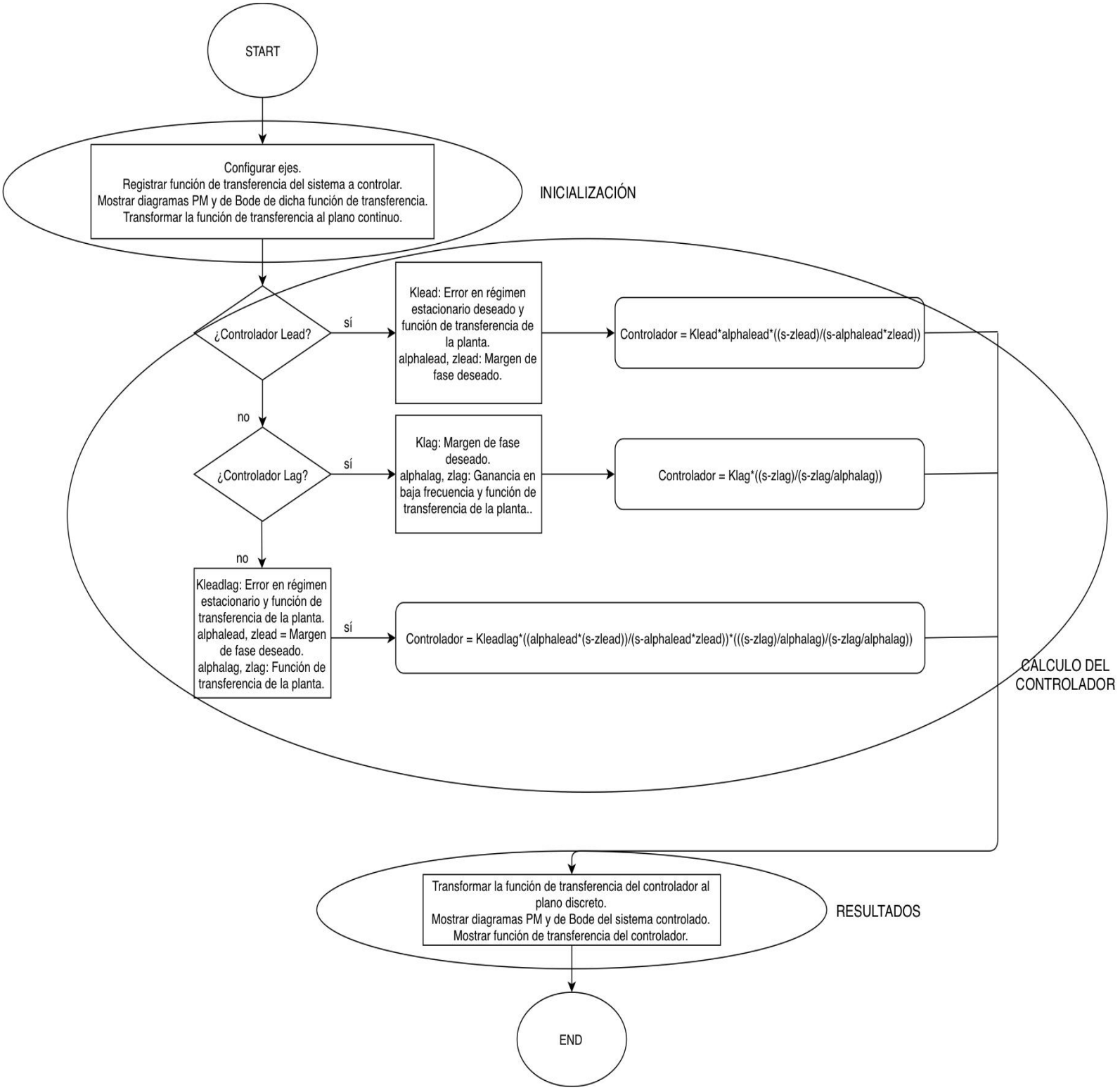

Diagrama de flujo del cálculo de controlador optimizado con DE (Punto 3.5)